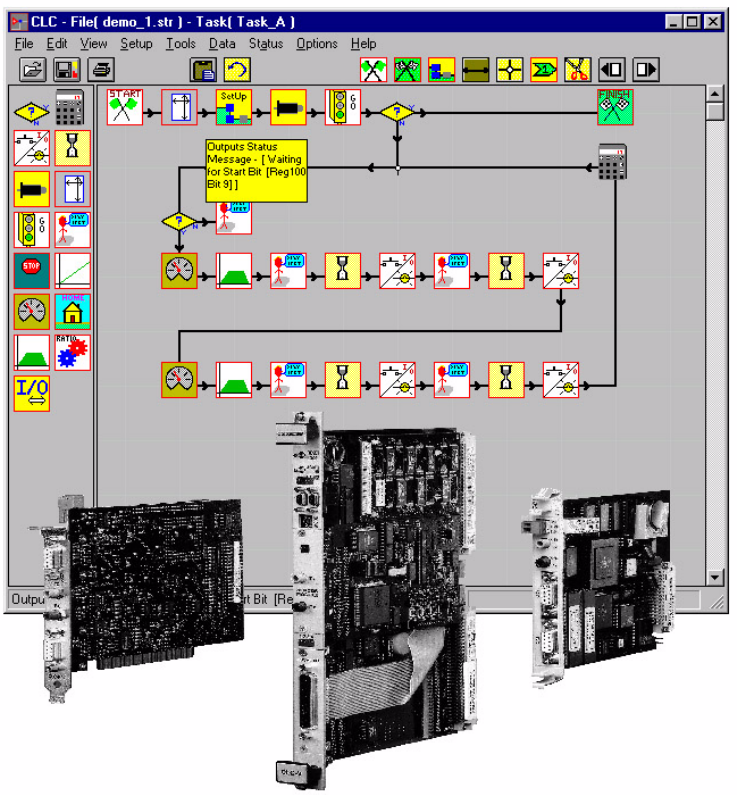
engineering
mannesmann
Rexroth



VisualMotion GPS 6.0

Startup Guide

Rexroth
Indramat

Title VisualMotion 6.0

Type of documentation Startup Guide

Document typecode DOK-VISMOT-VM*-06VRS**-PR02-AE-P

internal filing remarks

- Publication number
  120-2300-B302-02/AE

Purpose of this document This document describes ....

- the VisualMotion System components
- the required setup of CLC cards
- VisualMotion programming examples
- DDE Client interfaces for non-Indramat products
- Appendices outlining use of PC interfaces and Demo programs
- Fieldbus Supplements covering Profibus, Interbus and DeviceNet

Record of revisions

Revision	Date	Remarks
01	06/99	Official release for version 6
02	09/99	Updated Fieldbus Supplements

Copyright © INDRAMAT GmbH, 1999

Copying this document, and giving it to others and the use or communication of the contents thereof without express authority, are forbidden. Offenders are liable for the payment of damages. All rights are reserved in the event of the grant of a patent or the registration of a utility model or design (DIN 34-1).

Validity All rights are reserved with respect to the content of this documentation and the availability of the product.

Published by INDRAMAT Hoffman Estates • 5150 Prairie Stone Parkway • Hoffman Estates, IL 60192

Phone 847-645-3600 • Fax 847-645-6201 •

Dept. TechDoc (DP, HK)

Contents

1 Introduction 1-1

1.1 Purpose of Manual ... 1-1
Manual Overview ... 1-2
1.2 VisualMotion Overview ... 1-3
VisualMotion System Components ... 1-3
CLC System Architecture ... 1-9
Indramat's VisualMotion® Programming Interface ... 1-10
BTC06 ... 1-10
CLC Operating System ... 1-10
1.3 VM System Motion Capabilities ... 1-12
Non-Coordinated Motion ... 1-12
Coordinated Motion ... 1-12
Electronic Line Shaft (ELS) ... 1-13

2 System Installation 2-1

2.1 Introduction ... 2-1
Host System Requirements ... 2-1
Hardware Requirements ... 2-1
2.2 CLC Card Installation and Setup ... 2-3
Installing and Configuring the DSS SERCOS communication card ... 2-3
Installing and Configuring the CLC-D Motion Control Card ... 2-5
Installing and Configuring the CLC-P01 Motion Control Card ... 2-8
Installing and Configuring the CLC-P02 Motion Control Card ... 2-14
Upgrading CLC-P02 flash firmware ... 2-18
Installing and Configuring the CLC-V Motion Control Card ... 2-21
Upgrading CLC-V flash firmware ... 2-25
Saving CLC-V User Programs and Parameters to Flash Memory ... 2-27
Restoring CLC-V User Programs and Parameters from Flash Memory ... 2-27
Erasing CLC-V User Programs and Parameters from Flash Memory ... 2-28
2.3 VisualMotion Toolkit Installation and Setup ... 2-29

3 Create/Download Sample Program 3-1

3.1 Introduction ... 3-1
3.2 Sample Program ... 3-1
Homing Procedure ... 3-4
Save, Compile and Download Program ... 3-9
3.3 Program Variables ... 3-11
Assigning Labels to Variables ... 3-12

Assigning a Value to a Variable 3-15
3.4 File Types 3-16

4 Program Execution 4-1

4.1 Introduction 4-1
4.2 Initial Setup Prior to Operation 4-1
Control Registers 4-2
Parameter Mode 4-2
I/O Mapper 4-3
Bit Labels 4-7
Creating an I/O Map for a DEA Card 4-8
4.3 Run Sample Program 4-9
Program Operation 4-11
4.4 Parameter Overview 4-12
Viewing Parameters 4-13
Editing Parameters 4-13
Building Custom Parameter Lists 4-14

5 DDE Client Interfaces 5-1

5.1 Creating and Customizing a DDE Client Interface in Microsoft Excel™ 5-1
DDE Worksheet Functions 5-1
DDE Functions using Visual Basic® for Excel™ 5-4
5.2 Wonderware 5-8
Tagnames 5-10

A PC Interface (CLC-P: ISA - PC/104 bus) A-1

A.1 General information A-1
A.2 Base Address A-1
A.3 Bus Communications A-1
Dual Port RAM Organization A-2
A.4 Interrupts A-4

B VME (Versa Module Eurocard) Interface (CLC-V) B-1

B.1 CLC-V System Memory Organization B-1
CLC-VME Memory Window B-2
B.2 CLC-V Backplane Communication B-4
CLC-V Direct Data Access B-5
VME Memory Access Limitations B-5
B.3 VME Configuration B-7
VME Address and Access Modes B-7
VME System Signals B-8
VME Bus Parameters B-11

C Non-CLC Hardware C-1

C.1 VME Card Cage C-1
VME Card Cage Secondary Battery C-1

VME Backplane Jumpers ... C-3
Connecting AC Input Power ... C-3
C.2 CLC VME I/O Systems ... C-4
DEA I/O (DIAX Drive-Resident I/O) ... C-4
Xycom XVME-201 Digital I/O ... C-8
Xycom XVME-202 (PAMUX™ I/O) ... C-14
Xycom XVME-244 Digital I/O ... C-18
Western Reserve SmartMux (PAMUX™) ... C-22
Pentland MPV922 VME Digital I/O ... C-22
C.3 PC/ISA I/O Systems ... C-22

D Example Programs D-1

D.1 Example 1: Sequencer Application ... D-1
Program Instructions ... D-2
Program Layout ... D-5
Run Sequencer Program ... D-13
Program Operation ... D-15
Program Icons ... D-18
D.2 Example 2: Cam Application ... D-21
Program Instructions ... D-22
Program Layout ... D-27
CAM Program Icons ... D-29
D.3 Example 3: Programmable Limit Switch (PLS) Application ... D-30
Program Instructions ... D-30
Program Layout ... D-32
PLS Program Icons ... D-33
D.4 ECODRIVE 3 PLS Function ... D-34
Configuring Signal Status Word ... D-34
Modifying a Parameter List ... D-35
Programming PLS ON/OFF Signal ... D-37
D.5 Demonstration Program ... D-38
Linear Motion Application Demos ... D-38
Setup ... D-38
VisualMotion Icon Language Program - Demo_1 (Single Axis) ... D-39
VisualMotion Icon Language Program - Demo_2 (Coordinated Motion) ... D-42

I Index I-1

Supplement A - Profibus FieldBus Interfaces

Supplement B - DeviceNet Fieldbus Interfaces

Supplement C - Interbus Fieldbus Interfaces

List of Figures

Figure 1-1: VisualMotion System components 1-3
Figure 1-2: DIAX03 drive family 1-4
Figure 1-3: Motors used with DIAX03 1-5
Figure 1-4: DIAX04 drive family 1-6
Figure 1-5: Motors used with DIAX04 1-7
Figure 1-6: ECODRIVE 3 drive family 1-8
Figure 1-7: CCD Box 1-8
Figure 1-8: Different CLC versions 1-9
Figure 2-1: Plug-in card placement 2-3
Figure 2-2: Drive Address Setting through DSS card 2-4
Figure 2-3: CLC-D02.3M Hardware 2-5
Figure 2-4: Fiber optic ring structure 2-6
Figure 2-5: CLC-P01.1 Hardware 2-8
Figure 2-6: CLC-P01 memory allocation 2-10
Figure 2-7: CLC-P01 Jumper Location 2-11
Figure 2-8: Card Selection Setup 2-12
Figure 2-9: CLC-P01 Installation 2-13
Figure 2-10: CLC-P02.2 Hardware 2-14
Figure 2-11: CLC-P02 Jumper Locations 2-15
Figure 2-12: Card Selection Setup 2-16
Figure 2-13: CLC-P02 Installation 2-17
Figure 2-14: CLC-V02.3 Hardware 2-21
Figure 2-15: CLC-V Address Switch 2-22
Figure 2-16: CLC-V Slide Switch Location 2-22
Figure 2-17: Card Selection Setup 2-23
Figure 2-18: CLC-V Installation 2-24
Figure 2-19: CLC-V Installation Screens 2-33
Figure 2-20: CLC-D Installation Screens 2-34
Figure 2-21: CLC-P01 Installation Screens 2-35
Figure 2-22: CLC-P02 Installation Screens 2-36
Figure 3-1: Sample Program Icons 3-1
Figure 3-2: Completed sample program 3-8
Figure 3-3: Assigning a variable 3-14
Figure 3-4: Viewing and editing variables 3-15
Figure 4-1: Sample VisualMotion Toolkit Program 4-1
Figure 5-1: Excel Worksheet 5-1
Figure 5-2: Example DDE Formula 5-2
Figure 5-3: Example DDE Formula Result 5-2
Figure 5-4: Macro 1 5-5
Figure 5-5: Macro 2 5-7
Figure 5-6: DDE Access Name 5-8
Figure 5-7: Tagname for Displaying Parameters 5-10
Figure 5-8: Tagname for Writing to Variables 5-11
Figure A-1: Dual Port RAM Handshaking Protocol A-2
Figure B-1: CLC-V Memory Organization B-1
Figure B-2: VME Memory B-2
Figure B-3:Communication synchronization mailbox flags B-3
Figure B-4: VME bus Data Transfer Operations B-6
Figure C-1: VME Card Cage Secondary Battery C-1
Figure C-2: Replacement Battery for the VME Card Cage C-2
Figure C-3: Replacement Battery for the VME Card Cage C-3
Figure C-4: DEA Front Panel (4.2M, 5.2M, 6.2M) C-6
Figure C-5: XVME-201 Card - Jumper Locations C-12
Figure C-6: XVME-202 Card - Jumper Locations C-16
Figure C-7: XVME-244 Card - Jumper Locations C-20
Figure D-1: Task A - main sequencer program D-1
Figure D-2: MOVE_ABS subroutine D-6
Figure D-3: SET_IO subroutine D-7
Figure D-4: COUNT subroutine D-8

Figure D-5: WAIT subroutine D-9
Figure D-6: Main Sequencer - Part 1 of 3 ..... D-10
Figure D-7: Main Sequencer - Part 2 of 3 ..... D-11
Figure D-8: Main Sequencer - Part 3 of 3 ..... D-12
Figure D-9: Bit Labels D-13
Figure D-10: Arguments within a Function D-18
Figure D-11: CAM program - Part 1 of 2 ..... D-27
Figure D-12: CAM program - Part 2 of 2 ..... D-28
Figure D-13: programmable limit switch program D-30
Figure D-14: Programmable Limit Switch program D-33
Figure D-15: Configured Signal Status Word (S-0-0144) D-35
Figure D-16: Signal Status Word Configuration D-37

1 Introduction

1.1 Purpose of Manual

This manual is a start-up guide to the VisualMotion™ control system. VisualMotion is a combination of integrated digital multi-tasking motion control components, drives and motors used together to create a complete motion control solution. For information pertaining to digital drives and motors, refer to the following documentation:

- DIAX03 Drives with Electronic Gear Function, Documentation Set
  - DOK-DIAX03-ELS-04VRS**-50M1-EN-P, Material No. 273438
  - DOK-DIAX03-ELS-05VRS**-50M1-EN-P, Material No. 276238
- DIAX04 Drives with Electronic Gear Function, Documentation Set
  - DOK-DIAX04-ELS-05VRS**-60M1-EN-P, Material No. 276252
- ECODRIVE DKC02.1 SSE SERCOS Function, Documentation set
  - DOK-ECODRV-SSE-03VRS****-PRJ1-EN-P, Material No. 274825
- ECODRIVE DKC*.3 SMT Analog/SERCOS Functions, Documentation set
  - DOK-ECODR3-SMT-01VRS**-71M2-EN-P, Material No. 279086
- ECODRIVE DKC*.3 SGP ELS Functions, Documentation set
  - DOK-ECODR3-SGP-01VRS**-72M1-EN-P, Material No. 279094

The purpose of this manual is to provide a guideline for the first time user. The goal is not to develop the application program, but to assist the user in bringing the system on-line for the first time and insuring proper system functionality. The descriptions in this guide concerning drive displays, CLC card LED's, etc., assume that the reader is a first time user of the VisualMotion™ control system. It is assumed that the VisualMotion components and digital drives are installed properly and that all corresponding hard wiring has been completed. For more information, refer to the following VisualMotion Manuals:

- VisualMotion GPS 6.0, Trouble Shooting Guide
  - DOK-VISMOT-VM*-06VRS**-WAR1-AE-P, Material No. 282759
- VisualMotion GPS 6.0, Reference Manual
  - DOK-VISMOT-VM*-06VRS**-FKB1-AE-P, Material No. 280585

Manual Overview

Chapter 1 - Introduction	*Describes VisualMotion's components and the CLC's general theory of operation and its motion capabilities.*
Chapter 2 - System Installation	*Provides instructions on how to setup the CLC hardware and install the VisualMotion Toolkit (VMT)Windows based program.*
Chapter 3 - Create/Download Sample Program	*Provides instructions on how to create and download a sample program using VMT. It also discusses program variables and file types.*
Chapter 4 - Program Execution	*Includes a procedure for executing the sample program created and downloaded in Chapter 3, along with an I/O and Parameter overview.*
Chapter 5 - DDE Client Interfaces	*Includes examples on how to create custom DDE client interfaces for the CLC card using Excel and Wonderware.*
Appendix A - PC Interface (CLC-P:ISA - PC/104 bus)	*Covers setup and configuring of host computers for the following cards:* *CLC-P01 - personal or industrial PC with ISA connection* *CLC-P02 - personal or industrial PC with PC104*
Appendix B - VME Interface (CLC-V)	*Covers setup and configuring of the VME bus for the CLC-V02.3 - VME bus type used in card cages*
Appendix C - Non-CLC System Hardware	*Describes the Indramat VME card cage and CLC-supported I/O cards and subsystems.*
Appendix D - Example Programs	*Includes procedures to create and execute sample programs using the Visual Motion Icon programming environment. The sample programs cover Sequencer, Cam and Programmable Limit Switch (PLS) applications.*
Supplement A - Profibus Fieldbus Interfaces	• *General Profibus information and* • *Fieldbus Mapper examples* • *GPS programming information* • *PLC programming information* • *Profibus hardware*
Supplement B - DeviceNet Fieldbus Interfaces	• *General DeviceNet information and* • *Fieldbus Mapper examples* • *GPS programming information* • *PLC programming information* • *DeviceNet hardware*
Supplement C - Interbus Fieldbus Interfaces	• *General Interbus information and* • *Fieldbus Mapper examples* • *GPS programming information* • *PLC programming information* • *Interbus hardware*

1.2 VisualMotion Overview

The term VisualMotion is not only used to describe the Windows based program for creating motion control programs but also for all the components required for a complete motion control system. These components range from the CLC motion control card to the GPS firmware to the communications between the CLC card and all external components.

VisualMotion System Components

The VisualMotion system is composed of CLC control cards, GPS firmware *(software for the CLC cards)*, VisualMotion Toolkit (VMT) program, DDE Server *(communications protocol between windows programs and other applications),* BTC06 Handheld Teach Pendant (optional) and up to 40 intelligent digital drives all communicating over the SERCOS fiber optic ring.

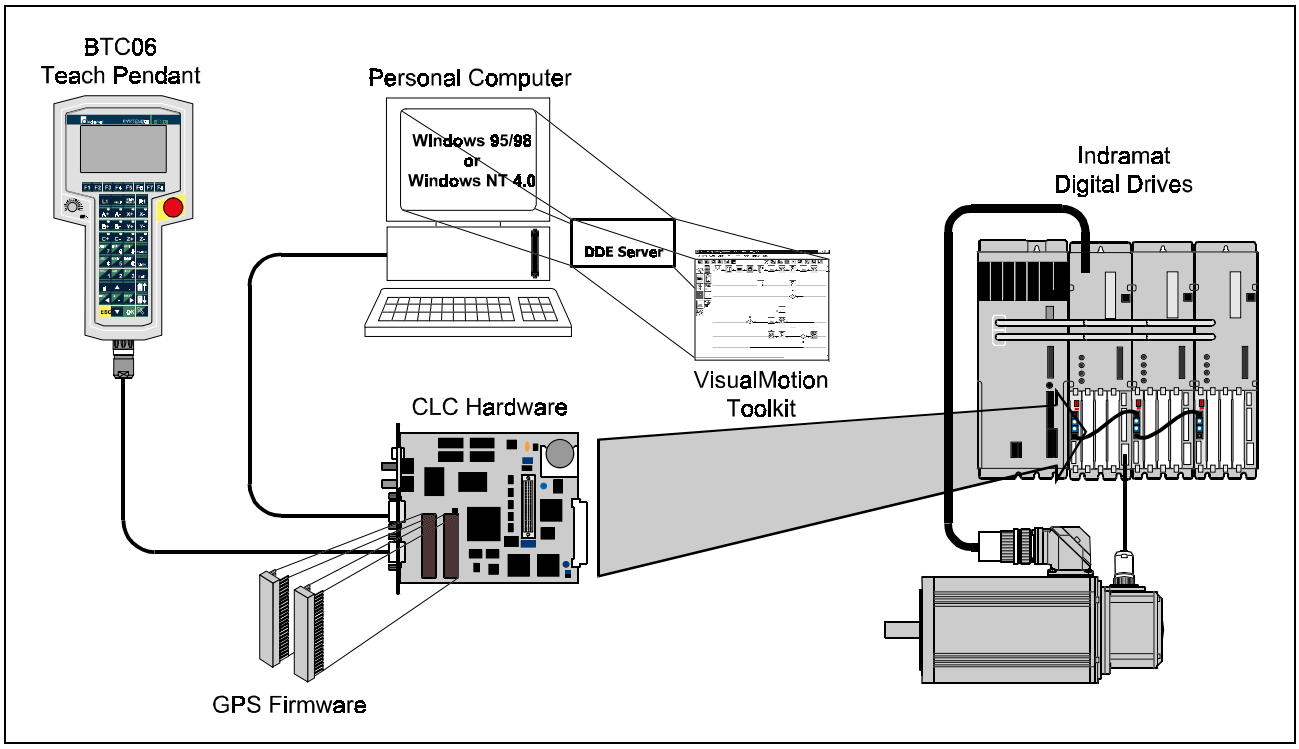


Figure 1-1: VisualMotion System components

Indramat's digital drives are made up of drive controllers and their associated motors. The digital drive families DIAX03, DIAX04 and ECODRIVE with CCD box are fully capable of using the functions available with the CLC motion control card.

DIAX03 digital drive controllers

The following digital drives make up the DIAX03 drive family.

- **DDS3.2:**
  Modular unit with two slots for plug-in cards and a continuous drive output of up to approximately 3 kW.
- **DDS2.2:**
  Modular unit with four slots for plug-in cards and a continuous drive output of up to approximately 12 kW.
- **DKR3.1:**
  Compact unit with four slots for plug-in cards and a continuous drive output of up to approximately 30 kW.
- **DKR2.1:**
  Compact unit with four slots for plug-in cards and a continuous drive output of up to approximately 50 kW.
- **DKR4.1:**
  Compact unit with four slots for plug-in cards and a continuous drive output of up to approximately 90 kW.
- **DKR5.1:**
  Compact unit with four slots for plug-in cards and a continuous drive output of up to approximately 224 kW.

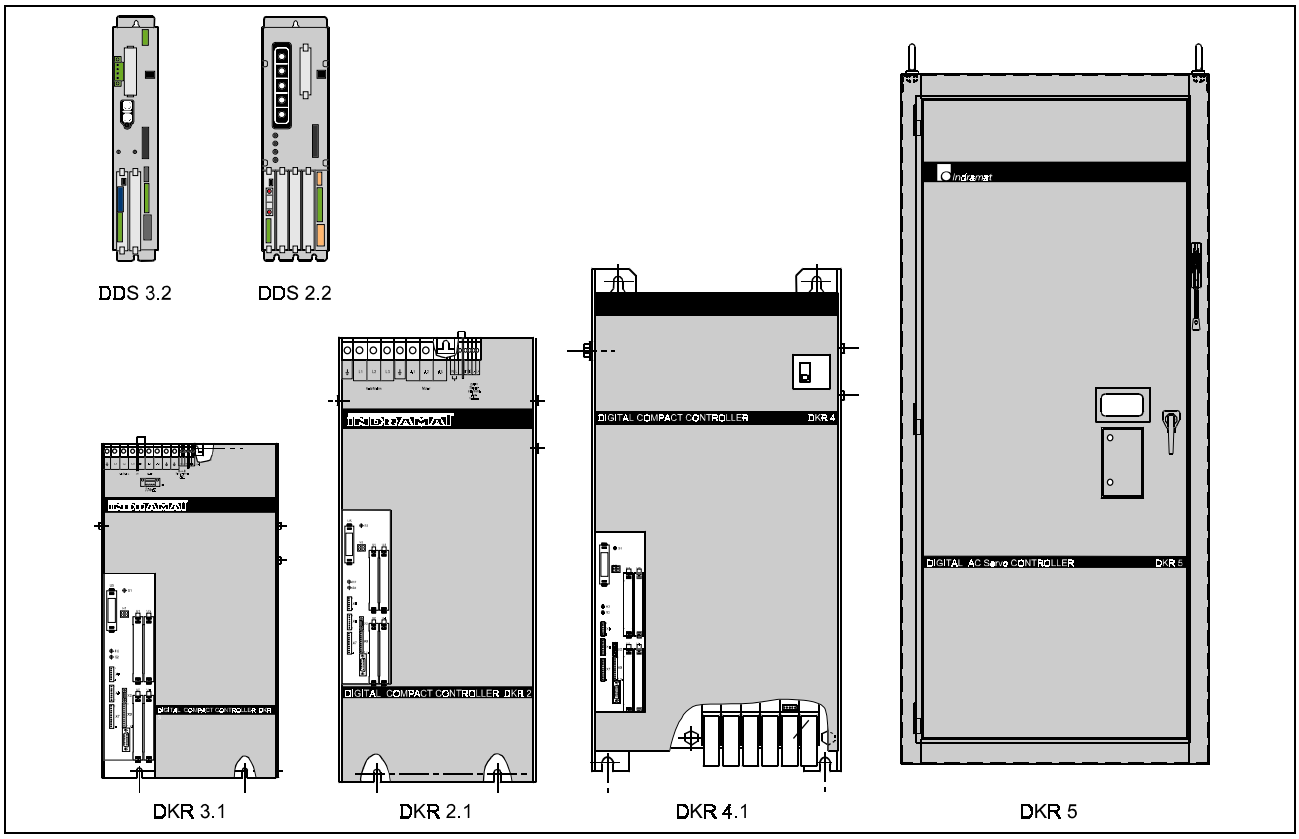


Figure 1-2: DIAX03 drive family

Motors used with DIAX03

All DIAX03 drive controllers are capable of operating all rotating and linear motors of the MDD, 2AD, 1MB, MBW, LAR and LAF series.

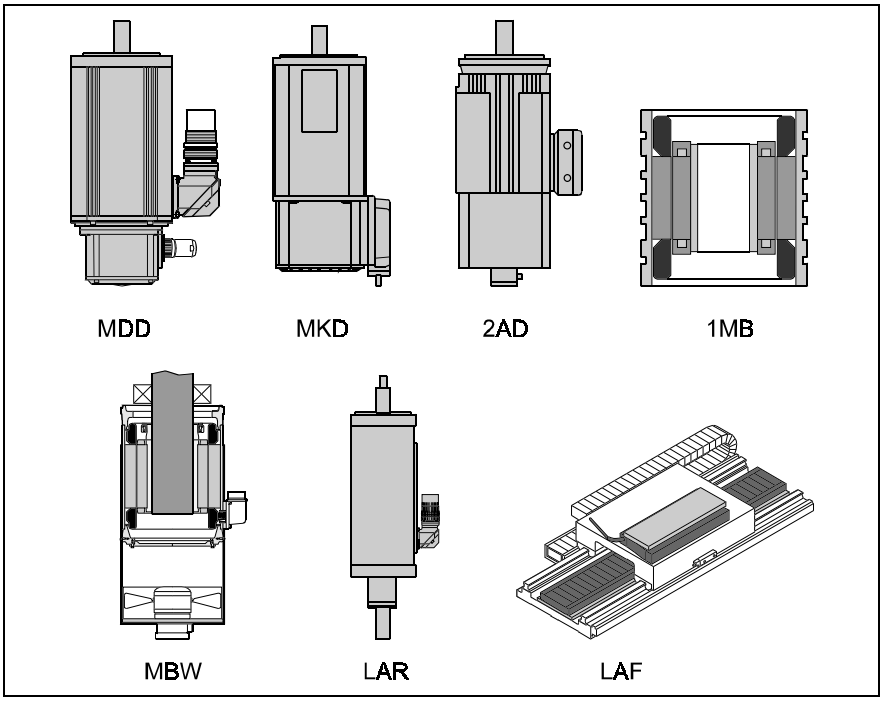


Figure 1-3: Motors used with DIAX03

DIAX04 digital drive controllers

The following digital drives make up the DIAX04 drive family.

- **HDD2.2:**
  A dual axes modular unit with two slots for plug-in cards per axis and a continuous drive output of up to approximately 2.5 kW.
- **HDS2.2:**
  Modular unit with four slots for plug-in cards and a continuous drive output of up to approximately 6.5 kW.
- **HDS3.2:**
  Modular unit with four slots for plug-in cards and a continuous drive output of up to approximately 15.5 kW.
- **HDS4.2:**
  Modular unit with four slots for plug-in cards and a continuous drive output of up to approximately 35 kW.

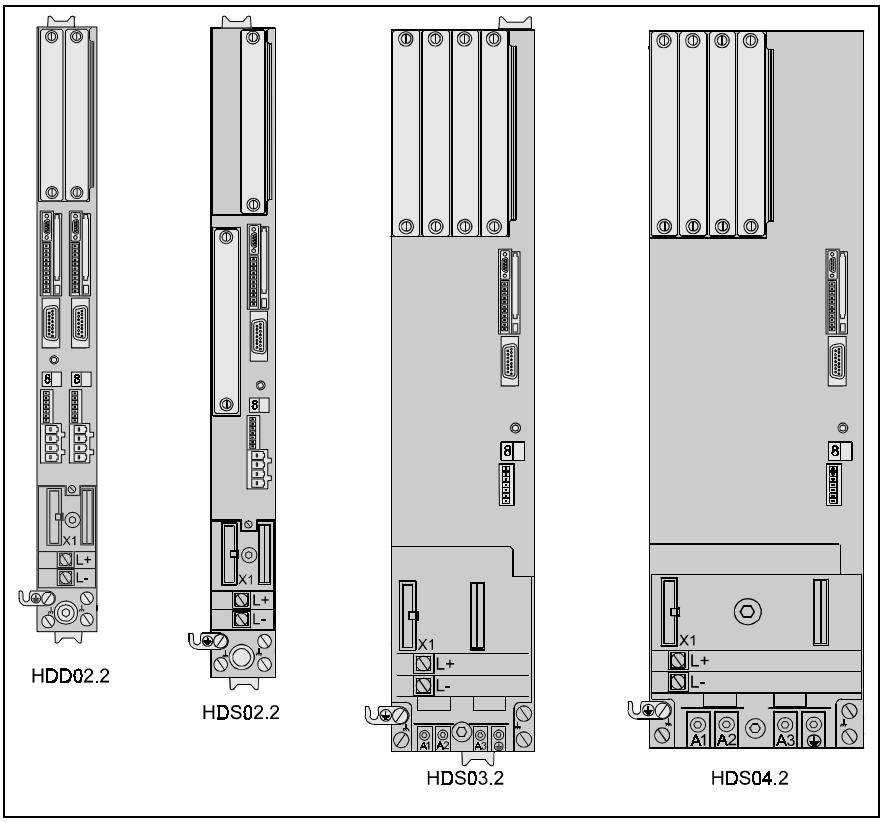


Figure 1-4: DIAX04 drive family

Motors used with DIAX04

All DIAX04 drive controllers are capable of operating all rotating and linear motors of the MHD, MKE, MKD, 2AD, 1MB, MBW, LAR and LAF series.

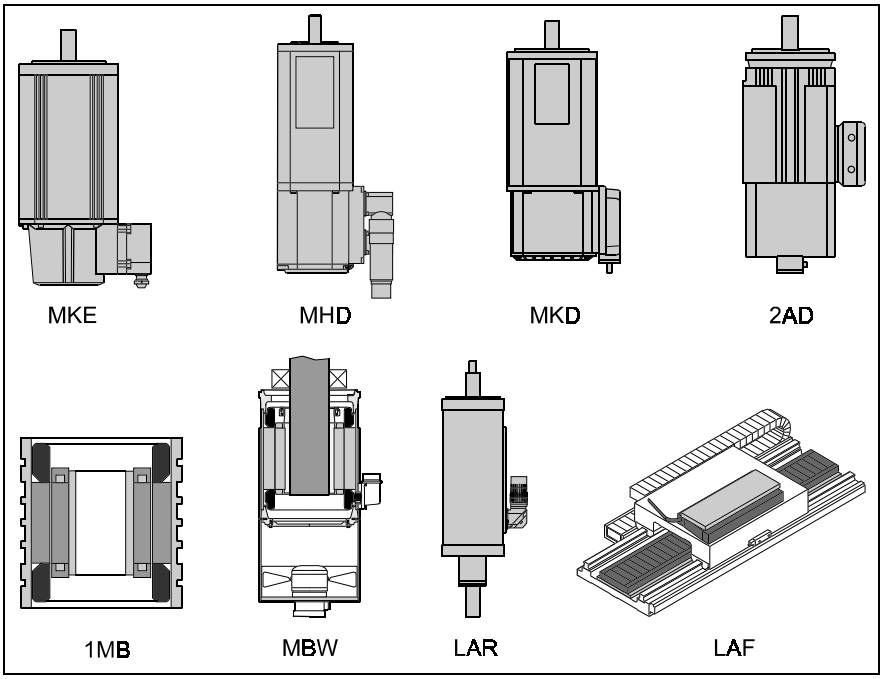


Figure 1-5: Motors used with DIAX04

ECODRIVE 3 digital drive controllers

The following digital drives make up the ECODRIVE 3 drive family. DKC drives have no slots available for plug-in cards; however, when combined with the stand alone CCD box, an ECODRIVE 3 system can have up to four slots available for plug-in cards.

- **DKCx.3-040:**
  can provide continuous drive output of up to approximately 1.5 kW.
- **DKCx.3-100:**
  can provide continuous drive output of up to approximately 4.0 kW.

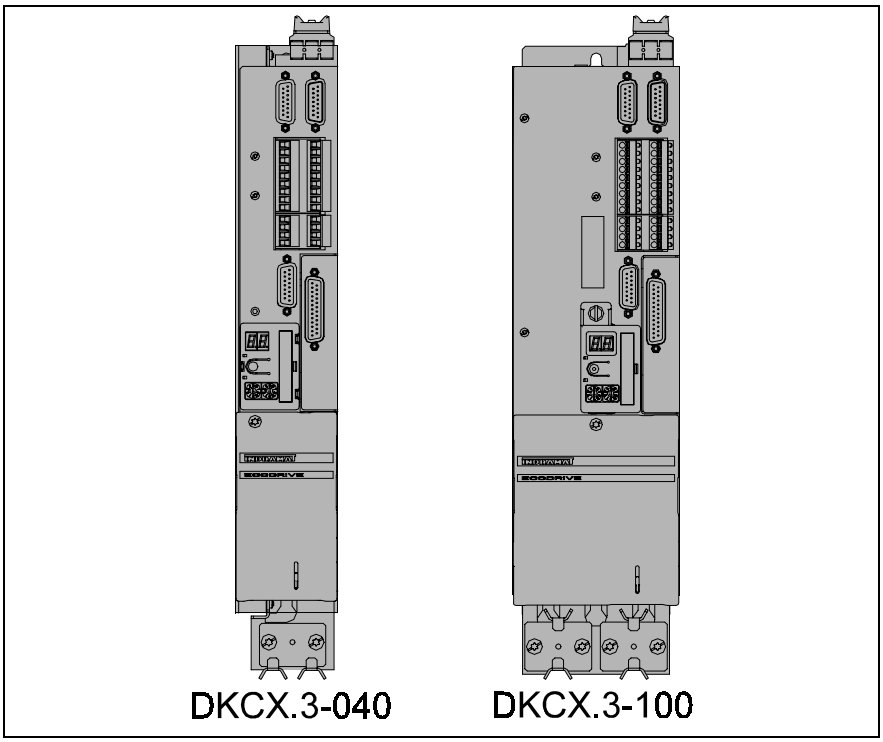


Figure 1-6: ECODRIVE 3 drive family

- **CCD Box:**
  provides 24V backplane power for CLC-D02.3 in combination with DEA 28/29/30 I/O cards.

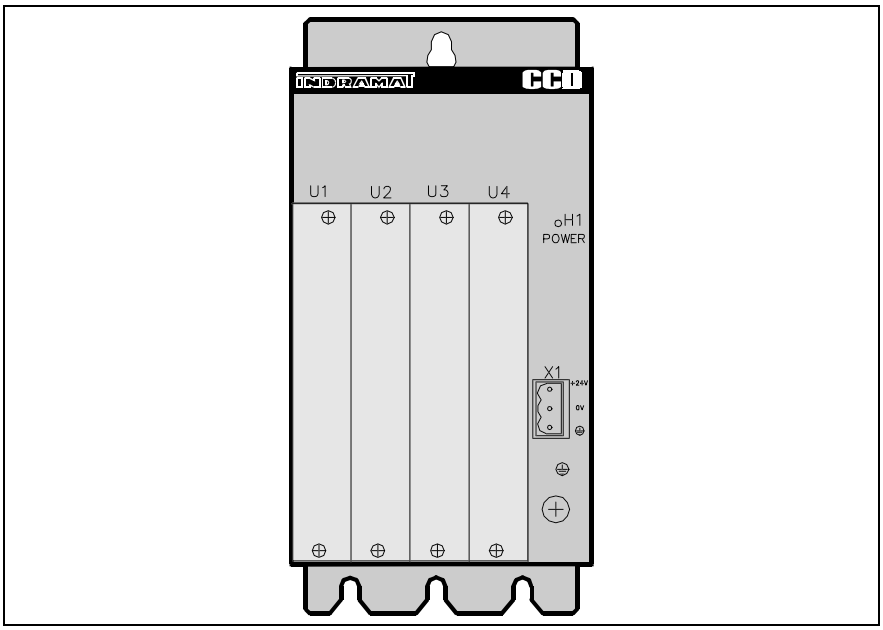


Figure 1-7: CCD Box

CLC System Architecture

The CLC card is a part of a larger motion control system which also includes digital servo drives and SERCOS, a fiber-optic communication system. The VM Controller can provide multi-axis coordinated and non-coordinated motion control with tightly integrated I/O logic control functions. The flexibility of the VM System allows it to be used for a wide variety of applications, from general motion control to sophisticated multi-axis electronic line shafting (ELS) to robotics.

CLC controls use SERCOS (**SE**rial **R**ealtime **CO**mmunications **S**ystem) fiber-optic interface to interconnect with Indramat drives. The SERCOS interface is an internationally accepted standard for real-time high-speed digital communication.

- requires only a single daisy-chained fiber-optic cable interconnecting the drives with the control
- Synchronous data protocol guarantees response time
- provides continuous monitoring and diagnostics for all devices and includes comprehensive standardized definitions for control loop functions
- High noise immunity and electrical isolation

The CLC-P/V open bus architecture provides easy I/O interfacing to virtually any manufacturer's PLC's and I/O cards. Both digital and analog I/O are available. Interrupt-type inputs can be used to provide minimum response time recognition of external events.

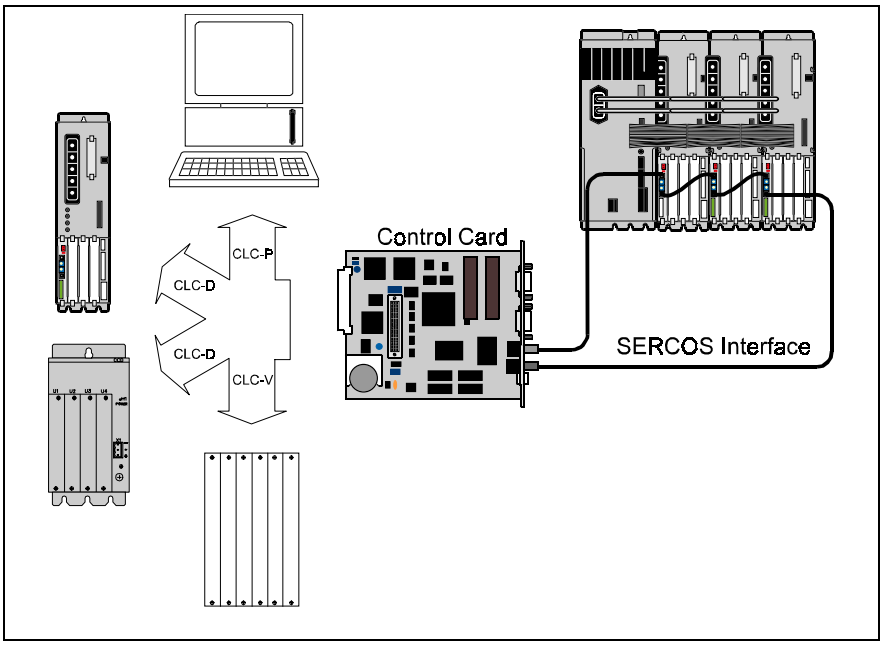


Figure 1-8: Different CLC versions

Four versions of the CLC card are currently available:

CLC-D

Plugs into Indramat's digital drives, providing an exceptional cost-effective motion control solution.

CLC-P01

IBM PC-AT bus architecture.

CLC-P02

PC/104 bus architecture.

CLC-V

VME bus architecture.

Indramat's VisualMotion® Programming Interface

The CLC motion control card combines an integrated multi-tasking environment with a unique graphical Windows based programming interface - VisualMotion ® (VM). VM provides simplified point-and-click programming, operation and management. With this software, system builders have a flexible and comprehensive environment, including easy DDE integration with applications such as Wonderware, InTouch or Visual Basic programs.

BTC06

An optional BTC06 can also be used to control CLC operation and adjust CLC parameters. The BTC06 is a hand-held instrument with 16 x 40 character display and a 48-key sealed membrane keypad. The pendant provides a convenient operation and position programming interface for Indramat CLC Motion Control. **Refer to the *VisualMotion 6.0 Trouble Shooting Guide* for more information.**

CLC Operating System

The VM Controller can simultaneously control up to four independent user tasks (A, B, C & D). Each task can control a coordinated group of two or three axes and any number of independent motion axes. Depending on the application, a single CLC can control up to 40 axes.

Tasks

VisualMotion can have up to 4 tasks running in each program. Tasks A-D run simultaneously and are given equal priority. A task is a process that the user runs in his machine. Using VisualMotion, the user can have 4 separate processes or task running simultaneously and each task can be independent of each other.

Multitasking is cooperative and can consecutively execute one instruction from each task. Special event functions preempt normal multitasking for real-time response and are prioritized according to the hierarchy of the event's associated task (Task A - highest priority, Task D - lowest priority). Two additional executive tasks manage the user interfaces through serial communication ports. One controls communication to the Host PC, the other controls the BTC06 when used.

Each task also has a queue for events, permitting asynchronous initiation of multiple independent events within each task. VM Controller events are a privileged form of subroutine, suspending all program tasks until pending events are done. Each task may have up to four events active at one time. The CLC provides several event types including time-based, distance-based, and position-based events, and others.

Events

Events are basically interrupt driven subroutines. They can be triggered by a variety of methods, such as transition of an input, repeating timer, position trigger, etc.

The Event system is pre-emptive, giving task A events the highest priority, while events associated with Task D have the lowest priority. Therefore, an event currently being executed within Task C will be suspended by an event from Task B. Once the active events are completed the suspended program tasks automatically re-activate and resume execution.

1.3 VM System Motion Capabilities

The VM Controller supports three kinds of motion (non-coordinated, coordinated, and electronic line shaft) and several modes of each type.

Non-Coordinated Motion

Non-coordinated motion is primarily used to control a single independent axis. There are two modes of non-coordinated motion: Single Axis, for linear positioning required to achieve point-to-point movement, and Velocity Mode, as might be used for some spindle motor drives.

Single axis

Single axis motion commands within a user program are interpreted by the VM Controller and sent to the DDS drive. The user program tells the DDS drive the speed, and/or distance and acceleration it should use to internally develop a velocity profile, which is then maintained and controlled within the intelligent DDS drive. Consequently, single axis motion does not require continuous calculation by the VM Controller and consumes minimum CPU resources.

Velocity Mode

Velocity mode controls only the speed of the axis, without any position information. The intelligent DDS-2 drive maintains torque and velocity loops internally, updating the internal loops every 250 microseconds.

Ratioed Axes

A special form of non-coordinated motion permits linking two axes by relating the number of revolutions of a slave axis to a master axis. For example, a ratio might be required when the positioning axis of a gantry robot, having a motor on each side of its supporting track, must travel along a circular track.

Coordinated Motion

The VM Controller defines multi-axis coordinated motion in terms of a path composed of standard straight line and circular geometry segments. Point positions, (x, y, z), are used to establish the start, middle or end of a geometry segment. Two points define a line, three points define a circle. The path combines these standard geometry segments so that the start of the next segment begins at the end of the previous segment. A path, therefore, is nothing more than a collection of connected segments.

Since each segment has an end point specifying speed, acceleration, deceleration and jerk, each segment can have a unique rate profile curve. A special type of segment, called a blend segment, can be used to join two standard geometry segments. Blend segments provide the capability of continuous smooth motion from one standard segment to another without stopping. They reduce calculation cycle time as well as provide a means of optimal path shaping.

The VM System is capable of calculating a path in any of several different modes:

Constant Speed

Constant Speed mode is always active and tries to maintain a constant speed between any two connecting segments in the path. This mode is constrained by the system's acceleration and deceleration. Constant speed is the optimum path motion for applying adhesives or paint, and welding and some forms of cutting such as laser or water-jet, etc.

Linear Interpolation

A coordinated motion straight line segment is defined by two points. The motion is calculated from the end point of the last segment, or the current position if the system is not in motion, to the new end point. Multi-axis coordinated motion is used when a relationship must be maintained between two or more axes during motion.

Circular Interpolation

A coordinated motion circular segment is defined by three points. Circular motion begins with the end point of the last segment executed, or the current system position if the system is not in motion, moves in a circular arc through an intermediate point, and terminates at the specified endpoint.

Kinematics

In addition to the standard linear and circular segments, the VM Controller has the capability of executing forward and inverse kinematic movement by using an application-specific library of kinematic functions.

Kinematics must be developed by Indramat to customer specifications. Contact Indramat Applications Engineering to inquire about applications which could benefit from kinematics.

Electronic Line Shaft (ELS)

An Electronic Line Shaft is used to synchronize one or more slave axes to a master axis. An ELS master can be a real or virtual axis. A real master can be another axis in the system, or an external feedback device such as an encoder. A virtual master is a command generated by the VM System. *(See ELS Icon, Chapter 6 of the VisualMotion 6.0 Reference manual)* Each slave axis can use either **velocity**, **phase** or **cam synchronization**. An ELS also includes the capability to jog each axis synchronously or independently, and to adjust phase offset and velocity while the program is running.

Velocity synchronization relates slave axes to a master in terms of rotational rate. It is used when axis velocities are most critical, as in paper processing operations in which two or more motors act on a single piece of fragile material.

Phase synchronization maintains the same relative position among axes, but adjusts the lead or lag of the slaves to the master in terms of degrees. It is used when the positions of axes are most critical. For example, to achieve proper registration in printing operations, the axis controlling the print head may be programmed for a particular phase offset relative to some locating device, such as a proximity switch.

Cam synchronization is used when custom position, velocity or acceleration profiles are needed at a slave axis. These special profiles are developed at the slave by sending position commands every SERCOS cycle.

A cam is an (x, y) table of positions that relate a master axis to a slave. Cams can be stored on the CLC card or on the digital drive. CLC cams have more adjustment options and can work with any SERCOS drive. Drive cams are more efficient and can be applied to more axes. The same programming commands and utilities are used for both drive-resident and card-resident cams.

See Appendix C - ELS Configuration of the VisualMotion 6.0 Reference manual for more information.

2 System Installation

2.1 Introduction

This chapter will instruct the user in the proper steps required for setting up the necessary VisualMotion® Hardware components as well as the installation of the VisualMotion Toolkit **(VMT)** Windows™ based program. The intent is to make the user familiar with the different versions of CLC hardware and the methods used for upgrading VisualMotion firmware.

Host System Requirements

Computer VisualMotion Toolkit **(VMT)** can be installed on any IBM™ PC compatible Pentium computer running…

- Windows95, Windows98 or Windows NT 4
- 4 Mb of RAM system memory
- Running Windows in enhanced mode is recommended
- Complete dual language (*English and German*) installation requires 16 MB of Harddisk space (3.5 MB is reserved for the help system). Additional space is required for user files.

Display A VGA display is required. A color display allows you to take full advantage of VMT's graphic interface.

Printer VMT uses the default printer installed on your computer. For optimum resolution, specially when printing icon programs, use a high resolution (300 dpi) laser or ink jet printer.

Mouse A serial or bus mouse is required to use VMT's Icon programming environment.

Serial I/O VMT can be configured to use the PC's serial port for communication between the Host PC and the CLC. A special RS-232 serial cable is required between the Host PC and the CLC Host communication port. Hardware handshaking is not used.

Note: When VisualMotion Toolkit is configured for serial communication, it requires exclusive use of its assigned serial port (specifically the serial port interrupt). Some PCs have additional accessory or I/O cards that share interrupts, as when a card has been added for an internal modem, fax card, or third and fourth serial port. These other devices should not be used when running VisualMotion.

Hardware Requirements

CLC-D Installed in Indramat's Digital Drives and CCD Box (Indramat's stand alone digital card cage)

CLC-P01 Installed in a PC's ISA bus (8-bit)

CLC-P02 Installed in a PC's PC104 bus

CLC-V Installed on a (**V**ersa **M**odule **E**urocard) VME bus

Note: For more information see Appendix A or the VisualMotion Trouble Shooting Guide listed on page 1-1.

DIAX03 digital drives

Indramat digital drives ranging from 3 to 224 kW of continuous power. Refer to *Figure 1-2, DIAX03 drive family* for more information.

DIAX04 digital drives

Indramat digital drives ranging from 2.5 to 35 kW of continuous power. Refer to *Figure 1-4, DIAX04 drive family* for more information.

ECODRIVE 3 digital drives

Indramat digital drives ranging from 1.5 to 4.0kW of continuous power. Refer to *Figure 1-6, ECODRIVE 3 drive family* for more information.

Motor(s)

Indramat motors are sized based on the digital drive's continuous output power. Contact Indramat's Service department for assistance.

Drive Programming Module

FWA-DIAX03-SSE-02VRS-MS (**S**tandard **SE**RCOS)

FWA-DIAX03-ELS-05VRS-MS (**E**lectronic **L**ine **S**hafting)

FWA-DIAX04-SSE-02VRS-MS (**S**tandard **SE**RCOS)

FWA-DIAX04-ELS-05VRS-MS (**E**lectronic **L**ine **S**hafting)

FWA-ECODR3-SMT-01VRS-MS (Analog / Standard SERCOS)

FWA-ECODR3-SGP-01VRS-MS (SERCOS ELS)

Contact Indramat's Service department for assistance.

SERCOS Module

DSS 2.1 - Installed in digital drive slot U1

Fiber Optic Cables

Indramat's IKO982 Fiber Optic Cable is designed for SERCOS communication between plug-in cards.

Indramat's IKO985 Fiber Optic Cable is designed for SERCOS communication of drives and external devices not normally contained within an enclosure.

Serial Cables

Indramat IKS0110 RS-232 serial cable for communication between computer and CLC-V.

Indramat IKS0061 RS-232 serial cable for communication between computer and CLC-D, CLC-P01 or CLC-P02.

For information pertaining to the digital drives, refer to the documentation sets listed in chapter 1, page 1-1.

2.2 CLC Card Installation and Setup

In this section it is assumed that all the necessary VisualMotion® Components, such as digital drives and motors are configured properly and that all necessary power is available to the drives. The intent is to guide the user in the installation and setup of a new CLC card, containing no programs or parameters, and the VisualMotion Toolkit Windows based program.

Note: For more information, refer to the appropriate power supply and drive manuals.

Installing and Configuring the DSS SERCOS communication card

The DSS02.1M "SERCOS interface plug-in card" makes it possible to operate a digital drive with a SERCOS interface compatible control via a fiber optic cable. The DSS card also offers inputs for the evaluation of reference switches, travel range limit switches, probes and an E-STOP input.

Warning

Unexpected Movement of Machine Load

Damage to machine or injury to personnel can occur

⇒ Remove all motor connects to machine load before proceeding with installation.

Typically, the DSS SERCOS Interface card is installed in slot U1. Afterwhich, any additional plug-in cards, such as CLC and DEA, can be installed in any order. The following 2 axes digital drive setup illustrates a typically SERCOS ring layout containing DSS, CLC and DEA plug-in cards.

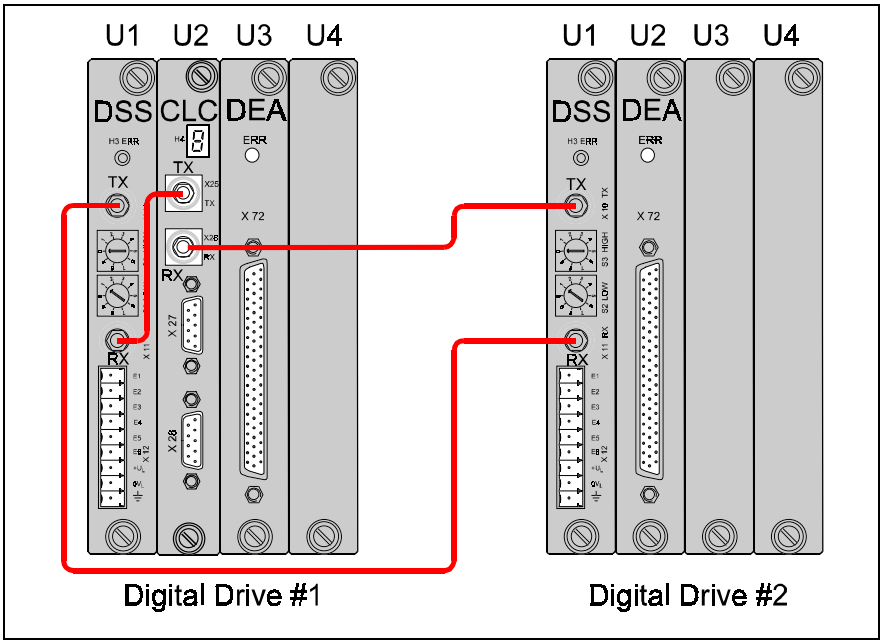


Figure 2-1: Plug-in card placement

Warning

Error during handling and mounting

Electro-static damage can result

⇒ Remove any power to the drives and accessories that are part of the system before inserting or removing any plug-in cards.

DSS drive address settings

In order to achieve proper drive addressing, set the S2 (low) and S3 (high) rotary selector switches on the DSS cards to a unique number for each drive contained in the SERCOS ring. For system above 9 axes, think of S2 (low) as the ones place and S3 (high) as the tens place in a two digit number.

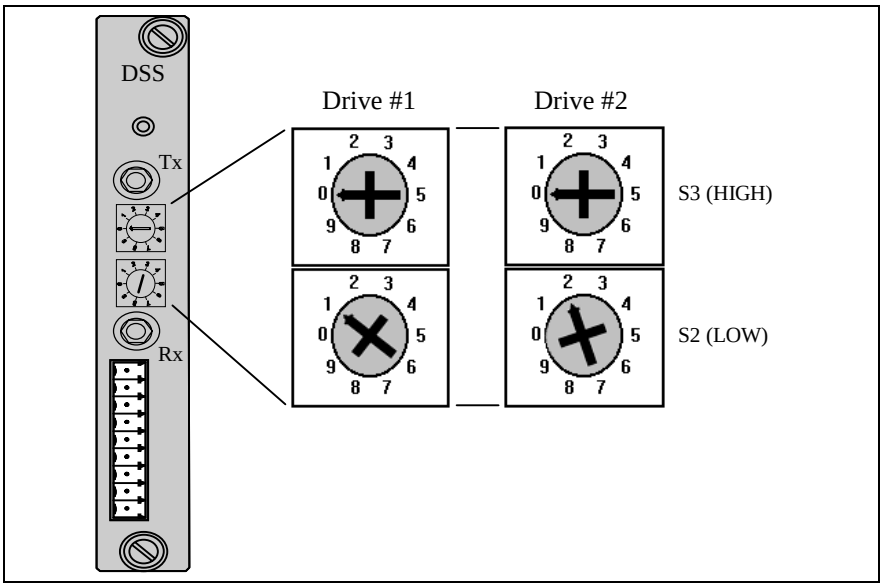


Figure 2-2: Drive Address Setting through DSS card

Installing and Configuring the CLC-D Motion Control Card

The CLC-D02.3M plug-in card is one of the hardware platforms used for motion control. The GPS (**G**eneral **P**urpose **S**oftware) firmware IC chips installed on this card contain the firmware and functionality of what makes up VisualMotion. To ensure proper installation and configuration of this essential component, follow the recommended steps contained in this section.

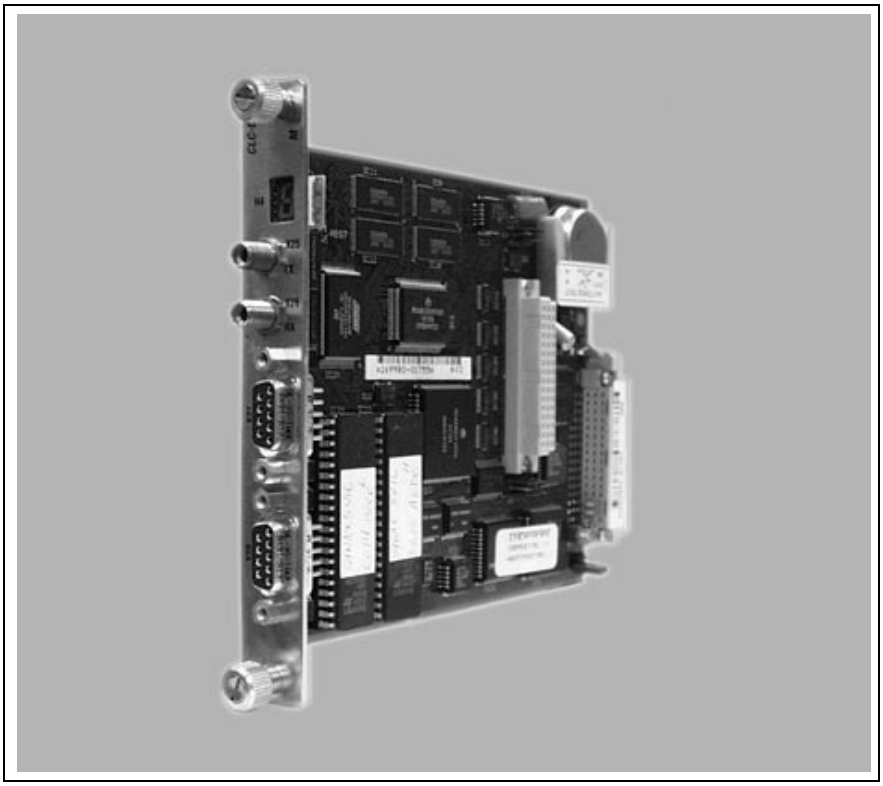


Figure 2-3: CLC-D02.3M Hardware

Caution

Electro-static discharge

Sensitive electronic device

⇒ The CLC card is a sensitive electronic device. Use caution when handling this board. Do not expose to Electro-static discharge or place the board directly on a conductive surface. Only handle the board by its face-plate or card edges.

Warning

Unexpected Movement of Machine Load

To reduce any possibility of unexpected movement during and after installation, remove all motor connects to machine load before proceeding.

⇒ Machine movement after installation should only be performed by trained personnel who understand the machine's specifications.

1. Before installing the CLC-D control card, reset the card's memory.
   - Remove the battery by sliding it out of the holder. Lay the battery down on a non-conductive surface.
   - Using an insulated wire with alligator clips, short the leads of capacitor C8 found below the battery.
   - Re-install the battery.

Warning

CLC-D Memory Reset

The procedure above resets the card's memory by removing the following items.

- CLC System parameters C-0-xxxx
- Axis parameters A-0-xxxx
- Task parameters
- CLC Cam tables, PLS and PID data
- Events, I/O Mapper, FieldBus Mapper
- Points tables, Variables, Zones
- Downloaded VisualMotion programs

2. Confirm the desired baudrate on connectors X27 and X28 before installing the card. Refer to the VisualMotion 6.0 Project Planning, for setting serial ports X27 and X28 via jumpers S1 and S2.

3. Install the CLC-D card in an available slot of the digital drive unit or the CCD stand alone card cage. Typically, the CLC-D card is installed next to the DSS card in slot U2.

4. Install the fiber-optic SERCOS ring by first connecting the CLC Tx to the DSS Rx of drive 1. Then connect the DSS Tx of drive 1 to the DSS Rx of drive 2. Repeat until the last drive (drive n) is connected. Then connect the DSS Tx of drive n to the CLC Rx to complete the ring.

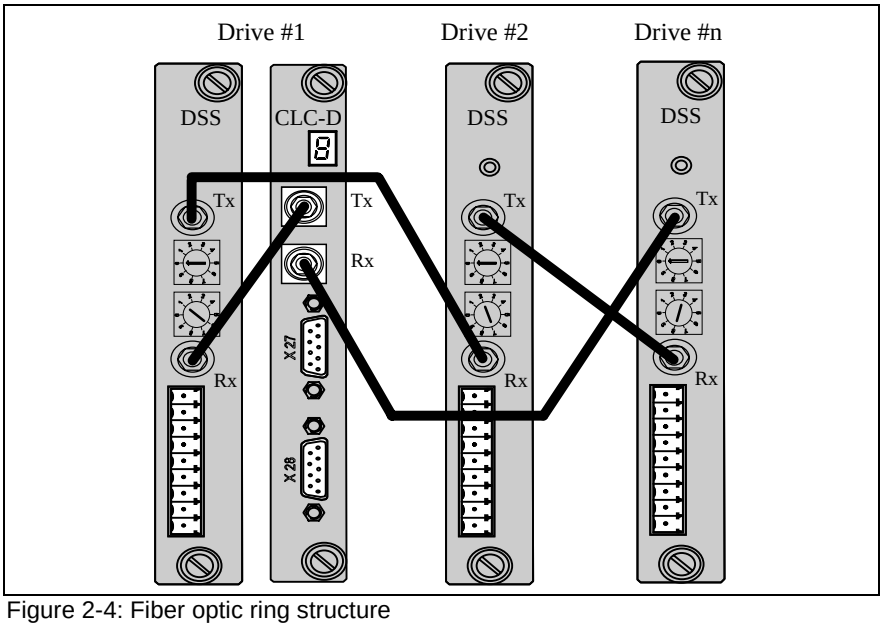


Figure 2-4: Fiber optic ring structure

5. Look for a red glow at the end of the cable from the last DSS's transmit port (Tx). This indicates a completed loop. If there is no glow, recheck the SERCOS connections at the DDS drive(s). If after verifying connections there is still no glow, one or more cables are suspect.

Warning

Fiber Optic Transmission Light

Bodily injuries

⇒ Do not expose your eyes to the red fiber optic transmission light. Confirm presence of light by reflecting it off your hand.

Note: For ECODRIVE units, the transmit (Tx) connector will illuminate a red glow regardless of a fiber optic connections from the CLC. Refer to the ECODRIVE's H1 display for verification of error free communications.

6. Connect the IKS0061 serial communications cable from the serial port of the Host PC to the serial port A (X27) of the CLC-D.

7. Power up the system.

8. The drive(s) will go through the SERCOS initialization stages. The communication process is divided into four communication phases. The CLC-D LED will display the phase codes and then the current status or error code.

   Example:
   Communication Phase 0
   Communication Phase 1
   Communication Phase 2
   Communication Phase 3
   Communication Phase 4

Since memory on the CLC card was cleared, an "E414 *Parameters were lost"* error code will be display on the CLC's 7-Segment LED display (H4.) This is an indication that the VisualMotion system is communicating properly.

To clear an *"E414"* error code, simply switch the drive system to and from parameter mode.

Note: Parameter mode switching occurs when Register 1, *System_Control,* bit 1 is set to 1. Register bits can be displayed within VisualMotion Toolkit under menu selection ***Data ⇒ Registers.***

After switching to and from Parameter Mode, the CLC will re-initialize and begin flashing an E400 *Emergency Stop*.

Note: E400 *Emergency Stop* occur when Register 1, *System_Control*, bit 3 is set to 0. To clear this error, set bit 3 to 1. In addition, bit 5, *Clear_All_Errors*, must also see a transition from 0 to 1 for the error to clear.

Installing and Configuring the CLC-P01 Motion Control Card

The CLC-P01 plug-in card uses an ISA bus interface. The card is designed for installation in personal or industrial PCs with an 8-bit ISA bus connection.

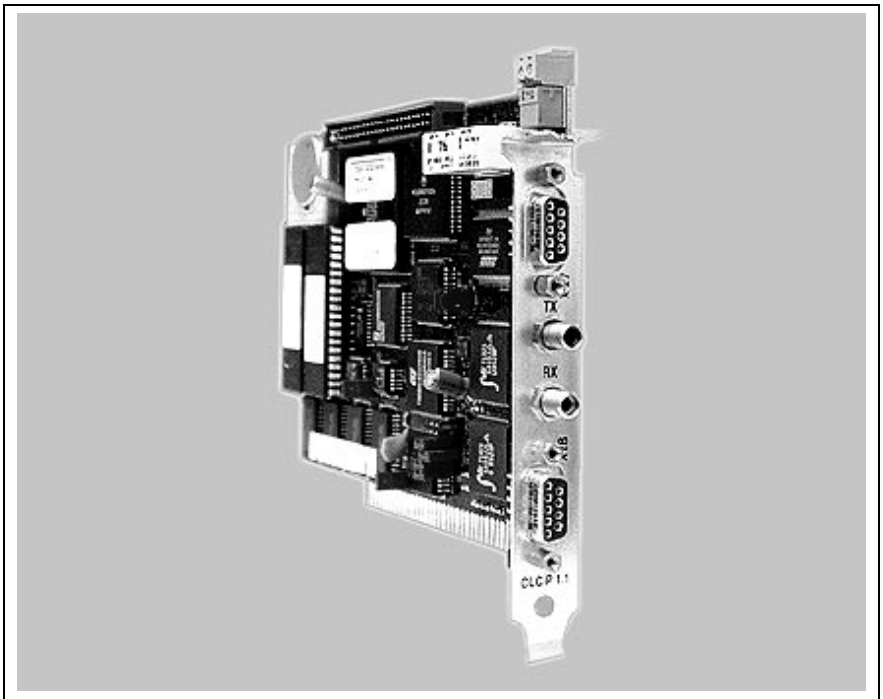


Figure 2-5: CLC-P01.1 Hardware

Caution

Electro-static discharge

Sensitive electronic device

⇒ The CLC card is a sensitive electronic device. Use caution when handling this board. Do not expose to Electro-static discharge or place the board directly on a conductive surface. Only handle the board by its face-plate or card edges.

Warning

Unexpected Movement of Machine Load

To reduce any possibility of unexpected movement during and after installation, remove all motor connects to machine load before proceeding.

⇒ Machine movement after installation should only be performed by trained personnel who understand the machine's specifications.

1. Before installing the CLC-P01 control card, reset the card's memory.
   - Remove the battery by sliding it out of the holder. Lay the battery down on a non-conductive surface.
   - Using an insulated wire with alligator clips, short the leads of capacitor C8 found below the battery.
   - Re-install the battery.

Warning

CLC-P01 Memory Reset
The procedure above resets the card's memory by removing the following items.

- CLC System parameters C-0-xxxx
- Axis parameters A-0-xxxx
- Task parameters
- CLC Cam tables, PLS and PID data
- Events, I/O Mapper, FieldBus Mapper
- Points tables, Variables, Zones
- Downloaded VisualMotion programs

2. Confirm the desired baudrate on connectors X27 and X28 before installing the card. Refer to Appendix A.2 - CLC-P01 Hardware for more information, for setting serial ports X27 and X28 via jumpers S1 and S2.

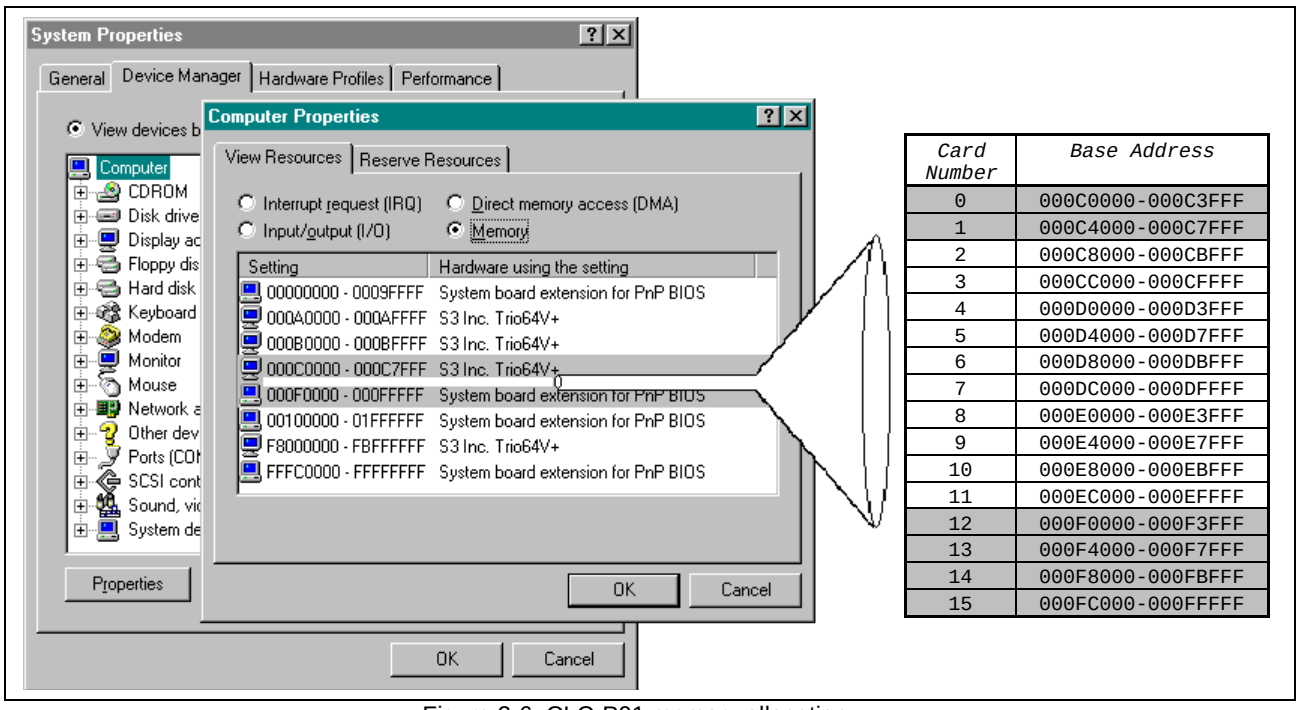
Windows 95/98 memory allocation

Before the CLC-P01 card can be installed, memory on the PC needs to be allocated to assure uninterrupted communications. Follow the procedure below prior to installing the CLC card into the PC.

3. Start Windows 95/98

4. From the Windows 95/98 toolbar select ***Start ⇒ Settings ⇒ Control Panel***

5. Double-click on the System icon . Select the Device Manager tab and click on the Properties button. The shaded area within the *Computer Properties* window in *Figure 2-6* represents the allowable memory range in which the CLC card communicates. Considering *Figure 2-6*, the possible memory locations are represented in the table as card number 2 through 11.



Card Number	Base Address
0	000C0000-000C3FFF
1	000C4000-000C7FFF
2	000C8000-000CBFFF
3	000CC000-000CFFFF
4	000D0000-000D3FFF
5	000D4000-000D7FFF
6	000D8000-000DBFFF
7	000DC000-000DFFFF
8	000E0000-000E3FFF
9	000E4000-000E7FFF
10	000E8000-000EBFFF
11	000EC000-000EFFFF
12	000F0000-000F3FFF
13	000F4000-000F7FFF
14	000F8000-000FBFFF
15	000FC000-000FFFFF

Figure 2-6: CLC-P01 memory allocation

6. Select an available card number and set jumpers S8 thru S11 on the CLC-P01 card to match the available base memory found in the previous step. Refer to *Figure 2-7: CLC-P01 Jumper Location*

PC Base Address selection (CLC-P01)

Card Number	*Base Address*	*S11*	*S10*	*S9*	*S8*
0	000C0000-000C3FFF	In	In	In	In
1	000C4000-000C7FFF	In	In	In	Out
2	000C8000-000CBFFF	In	In	Out	In
3	000CC000-000CFFFF	In	In	Out	Out
4	000D0000-000D3FFF	In	Out	In	In
5	000D4000-000D7FFF	In	Out	In	Out
6	000D8000-000DBFFF	In	Out	Out	In
7	000DC000-000DFFFF	In	Out	Out	Out
8	000E0000-000E3FFF	Out	In	In	In
9	000E4000-000E7FFF	Out	In	In	Out
10	000E8000-000EBFFF	Out	In	Out	In
11	000EC000-000EFFFF	Out	In	Out	Out
12	000F0000-000F3FFF	Out	Out	In	In
13	000F4000-000F7FFF	Out	Out	In	Out
14	000F8000-000FBFFF	Out	Out	Out	In
15	000FC000-000FFFFF	Out	Out	Out	Out

Table 2-1: CLC-P01 card number and base memory selection

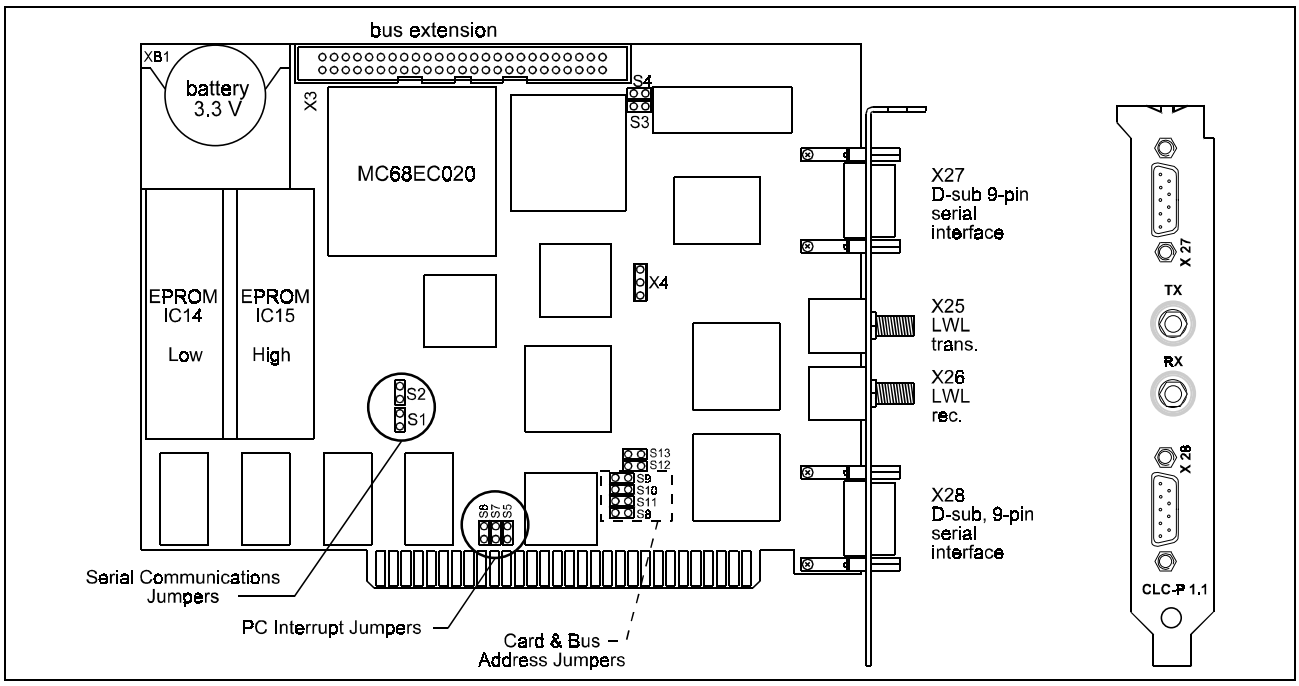


Figure 2-7: CLC-P01 Jumper Location

Windows NT Base memory setup

Unlike Windows 95/98, Windows NT automatically allocates memory resources for the devices installed in the PC. The base address setting on the CLC-P01 card is set identical to that of Windows 95. However, since the memory resources on Windows NT cannot be identified for verification of allocated memory, this process is slightly modified.

6. Select a base memory address for the CLC-P01 card from *Table 2-1: CLC-P01 card number and base memory selection*. Set jumpers S8 thru S11 to match the base memory address and continue to step 5.

Note: Based on Indramat's experience, begin with one of the base memory addresses for card numbers 4 thru 7. Theoretically, any of the 16 base memory addresses should work; however, not being able to verify memory allocation in Windows NT, Indramat has had to best success with card numbers 4 thru 7.

7. Now, set the PC IRQ interrupt. Jumpers S5 thru S7 set the interrupt line for the CLC to PC interrupt.

PC Interrupt selection (CLC-P01)

Interrupt	***IRQ Jumpers*** *(refer to figure 2-7)*		
	S5	***S6***	***S7***
IRQ2 (IRQ9)	In	Out	Out
IRQ3	Out	In	Out
IRQ5	Out	Out	Out

Table 2-2: PC Interrupt selection

Note: Select an Interrupt not currently use by the host computer.

8. Install the CLC-P01 card in the Host PC. Reconnect all PC connections and power up the PC.

Note: Proper connection of the CLC-P01 card is confirmed when the red fiber optic light is transmitted through ***Tx***.

Warning

Fiber Optic Transmission Light
Bodily injuries
⇒ Do not expose your eyes to the red fiber optic transmission light. Confirm presence of light by reflecting it off your hand.

9. If VisualMotion Toolkit **(VMT)** is installed on the PC containing the CLC-P01 card or a Laptop, continue to step 10. If VisualMotion is not installed, refer to section ***2.3 VisualMotion Toolkit Installation*** and Setup before proceeding.
10. Start VMT and choose ***Card Selection*** from the ***Setup*** menu. Select the appropriate connection method and set the card number to match the jumper settings of step 4.

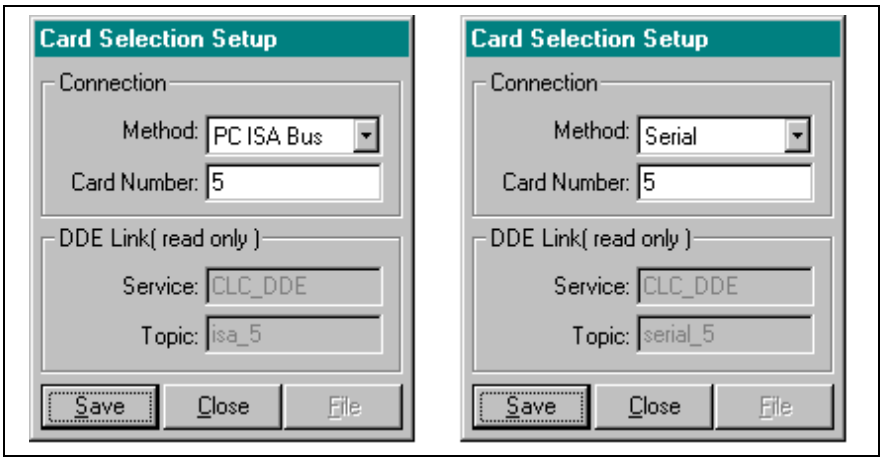


Figure 2-8: Card Selection Setup

Note: Proper VMT communication depends on the connection method selected.
- PC ISA Bus - communication between CLC-P01 and PC using ISA bus.
- Serial - serial communication between PC and CLC-P01 using com port.

11. To verify VMT communications with the CLC-P01 card, request a system status from menu selection ***Status ⇒ System***. If communications have been established, the *System Parameters* screen will appear containing information on the installed CLC card.

12. Once communications have been established, power down the Host PC and make the appropriate cable connections to the system as illustrated in *Figure 2-9: CLC-P01 Installation. Refer to Figure 2-4: Fiber optic ring structure* for details

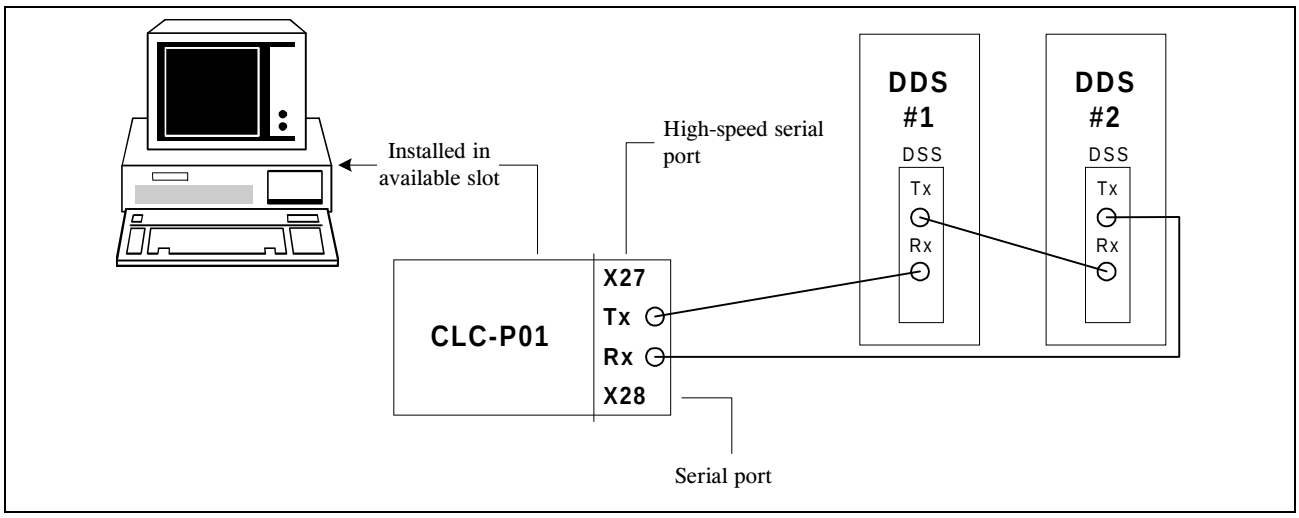


Figure 2-9: CLC-P01 Installation

13. Once the VisualMotion system is completely connected, power the Host PC and drive units for system initialization.

Since memory on the CLC card was cleared when the battery was removed, an "E414 *Parameters were lost"* error code will be present on the CLC. To view this error code, start VisualMotion Toolkit and select ***Status ⇒ System***. The *System Parameters* window will display this error code next to row heading *Diagnostic Message*. This is an indication that the VisualMotion system is communicating properly. To clear an *"E414"* error code, simply switch the drive system to and from parameter mode.

Note: Parameter mode switching occurs when Register 1, *System_Control,* bit 1 is set to 1. Register bits can be displayed within VisualMotion Toolkit under menu selection ***Data ⇒ Registers.***

After switching to and from Parameter Mode, the CLC will re-initialize and begin flashing an E400 *Emergency Stop*.

Note: E400 *Emergency Stop* occur when Register 1, *System_Control*, bit 3 is set to 0. To clear this error, set bit 3 to 1. In addition, bit 5, *Clear_All_Errors*, must also see a transition from 0 to 1 for the error to clear.

See Appendix A - PC Interfaces (CLC-P:ISA - PC//104 bus) for more information.

Installing and Configuring the CLC-P02 Motion Control Card

The CLC-P02 is a motion control card on a PC/104 platform. The VisualMotion firmware on this platform includes all of the features of the CLC-P01, with improvements in the configuration and memory capacity.

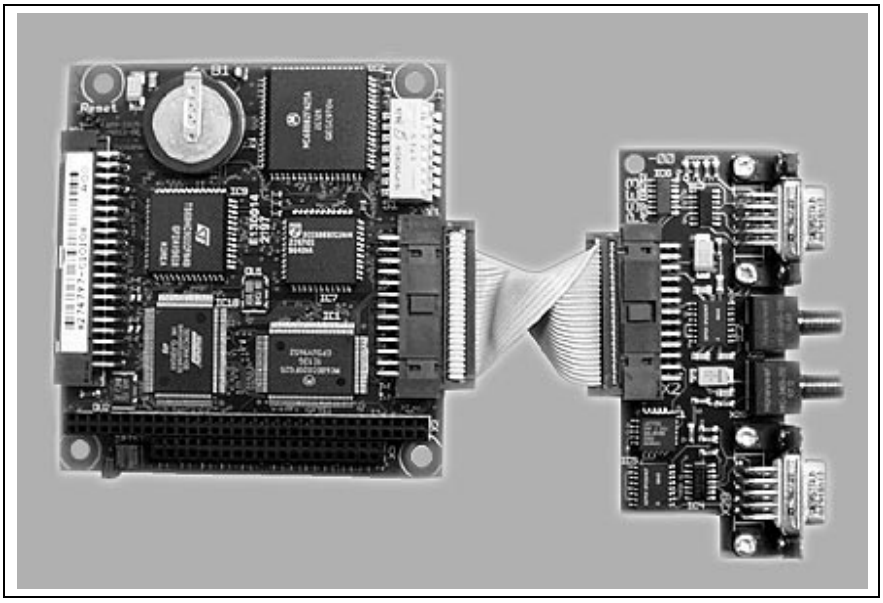


Figure 2-10: CLC-P02.2 Hardware

The CLC-P02 is a slave card that is typically installed onto a PC's or PLC's motherboard (master board) containing a PC/104 connector. After which, all the necessary connections for serial communications and fiber optics are made to the drive system.

Caution

Electro-static discharge
Sensitive electronic device
⇒ The CLC card is a sensitive electronic device. Use caution when handling this board. Do not expose to Electro-static discharge or place the board directly on a conductive surface. Only handle the board by its face-plate or card edges.

Warning

Unexpected Movement of Machine Load
To reduce any possibility of unexpected movement during and after installation, remove all motor connects to machine load before proceeding.

⇒ Machine movement after installation should only be performed by trained personnel who understand the machine's specifications.

1. Before the installation of the CLC-P02 card can begin, a memory address and a PC/104 interrupt must be set to assure uninterrupted communications. The setting of a base memory address and interrupt are made by opening and closing the DIP switch settings of S1. The following figure illustrates the location of the S1 switch on the CLC-P02.

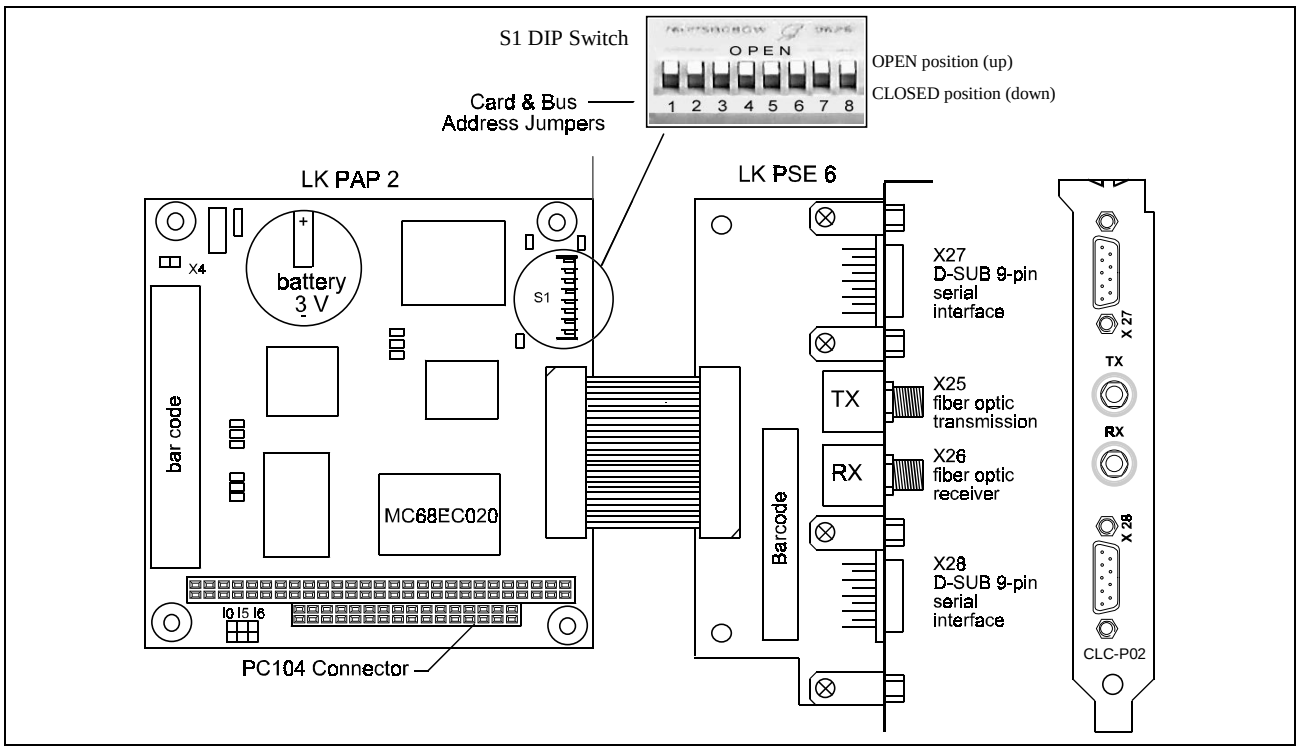


Figure 2-11: CLC-P02 Jumper Locations

2. Using the following table, set S1 DIP switches 1 thru 4 to an available base memory address. Refer to the CLC-P01 installation procedure on page 2-8 for PC memory allocation prior to setting the base memory address on the CLC-P02.

PC/104 Base Address selection (CLC-P02)

Card Number	*Base Memory Address*	*S1 DIP Switch setting* (refer to figure 2-11)			
		1	*2*	*3*	*4*
0	000D0000 - 000D1FFF	OPEN	CLOSED	CLOSED	CLOSED
1	000D2000 - 000D3FFF	OPEN	CLOSED	CLOSED	OPEN
2	000D4000 - 000D5FFF	OPEN	CLOSED	OPEN	CLOSED
3	000D6000 - 000D7FFF	OPEN	CLOSED	OPEN	OPEN
4	000D8000 - 000D9FFF	OPEN	OPEN	CLOSED	CLOSED
5	000DA000 - 000DBFFF	OPEN	OPEN	CLOSED	OPEN
6	000DC000 - 000DDFFF	OPEN	OPEN	OPEN	CLOSED
7	000DE000 - 000DFFFF	OPEN	OPEN	OPEN	OPEN
8	000E0000 - 000E1FFF	CLOSED	CLOSED	CLOSED	CLOSED
9	000E2000 - 000E3FFF	CLOSED	CLOSED	CLOSED	OPEN
10	000E4000 - 000E5FFF	CLOSED	CLOSED	OPEN	CLOSED
11	000E6000 - 000E7FFF	CLOSED	CLOSED	OPEN	OPEN
12	000E8000 - 000E9FFF	CLOSED	OPEN	CLOSED	CLOSED
13	000EA000 - 000EBFFF	CLOSED	OPEN	CLOSED	OPEN
14	000EC000 - 000EDFFF	CLOSED	OPEN	OPEN	CLOSED
15	000EE000 - 000EFFFF	CLOSED	OPEN	OPEN	OPEN

Table 2-3: PC/104 memory address selection (CLC-P02)

3. Now, set the PC/104 IRQ interrupt. Switches 5 thru 8 on the S1 DIP switch sets the interrupt line for the CLC to PC interrupt.

Note: Only one PC/104 interrupt switch can be in the closed position at one time; otherwise, an interrupt conflict will be encountered.

PC/104 Interrupt selection (CLC-P02)

Interrupt	***S1 DIP Switch setting*** *(refer to figure 2-11)*			
	5	***6***	***7***	***8***
INT15	CLOSED	OPEN	OPEN	OPEN
INT12	OPEN	CLOSED	OPEN	OPEN
INT11	OPEN	OPEN	CLOSED	OPEN
INT10	OPEN	OPEN	OPEN	CLOSED
None	OPEN	OPEN	OPEN	OPEN

Table 2-4: PC/104 Interrupt selection

4. Install the CLC-P02 card in the Host PC's PC/104 connector. Reconnect all PC connections and power up the PC.

Note: Proper connection of the CLC-P02 card is confirmed when the red fiber optic light is transmitted through ***Tx***.

Warning

Fiber Optic Transmission Light

Bodily injuries

⇒ Do not expose your eyes to the red fiber optic transmission light. Confirm presence of light by reflecting it off your hand.

5. If VisualMotion Toolkit **(VMT)** is installed on your PC containing the CLC-P02 card or a Laptop, continue to step 6. If VisualMotion is not installed, refer to section ***2.3, VisualMotion Toolkit Installation*** and Setup before proceeding.
6. Start VMT and choose ***Card Selection*** from the ***Setup*** menu. Select the appropriate connection method and set the card number to match the jumper settings of step 2.

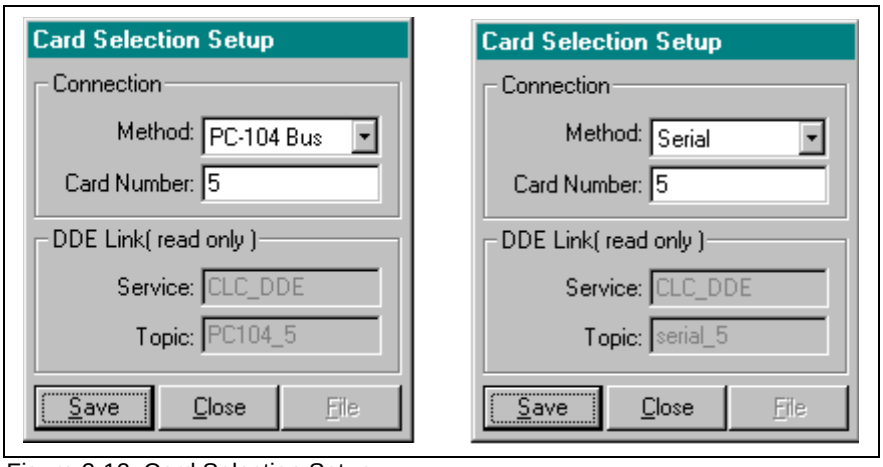


Figure 2-12: Card Selection Setup

Note: Proper VMT communication depends on the connection method selected.

- PC-104 Bus - communication between CLC-P02 and PC using a PC/104 bus.
- Serial - serial communication between PC and CLC-P02 using com port.

7. To verify VMT communications with the CLC-P02 card, request a system status from menu selection ***Status ⇒ System***. If communications have been established, the *System Parameters* screen will appear containing information on the installed CLC card.

8. Once communications have been established, power down the Host PC and make the appropriate cable connections to the system as illustrated in *Figure 2-13: CLC-P02 Installation. Refer to Figure 2-4: Fiber optic ring structure* for details.

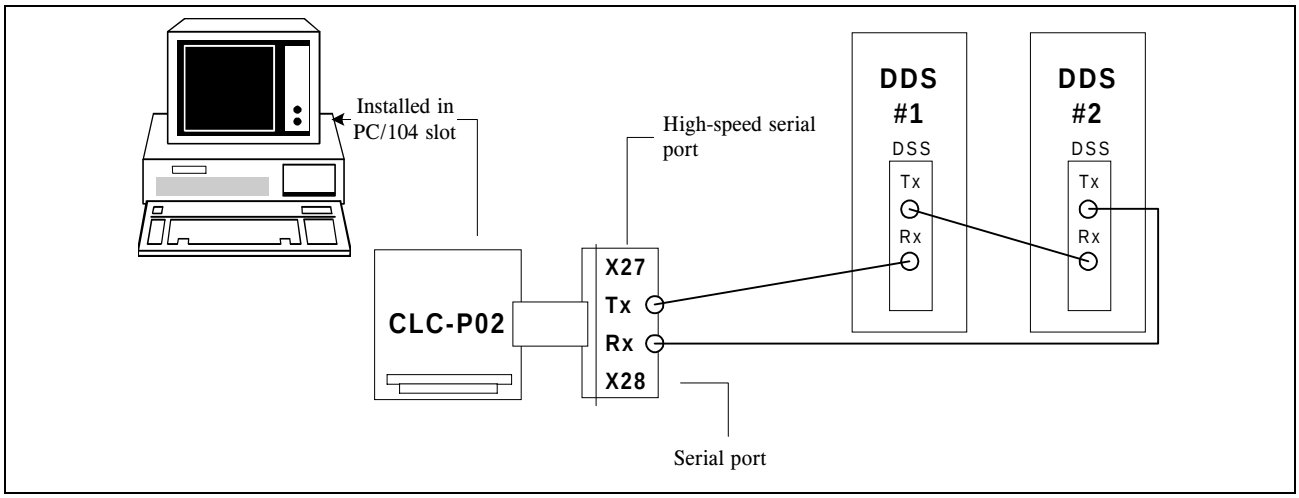


Figure 2-13: CLC-P02 Installation

9. Once the VisualMotion system is completely connected, power the Host PC and drive units for system initialization.

The system is now ready to be programmed. Refer to chapters 3 and 4 for programming details.

See Appendix A - PC Interfaces (CLC-P:ISA - PC//104 bus) for more information.

Upgrading CLC-P02 flash firmware

Perform a system archive to save all VisualMotion user programs and parameters before upgrading firmware.

Archiving VisualMotion System

Start VisualMotion Toolkit and select ***Archive*** from the *File* menu. Archive all programs and options, such as, cam tables, events, I/O mapper, etc, that apply to your application.

Warning

Loss of VisualMotion System

User programs will be loss when a firmware upgrade is performed.

⇒ The following is a list of items that may be loss during a firmware upgrade.

- CLC System parameters C-0-xxxx
- Axis parameters A-0-xxxx
- CLC Cam tables
- Events
- I/O Mapper; FieldBus Mapper
- Points tables, Variables
- Zones
- Downloaded VisualMotion programs

The CLC-P02 firmware can be upgrade without removing the card through the X27 serial port. The following procedure describes how to download a firmware file to the CLC, which is then burned to flash.

Serial Port Upgrade

Upgrading the CLC-P02 flash firmware using the X27 serial port requires the following:

- A PC running Windows 95/98 or NT4 with HyperTerminal installed.
- An IKS0061, RS-232 serial cable, is required for connecting to port A (X27) of the CLC.
- A firmware upgrade file named according to the hardware platform and firmware version. For example, *p2gps6v56.abs*.

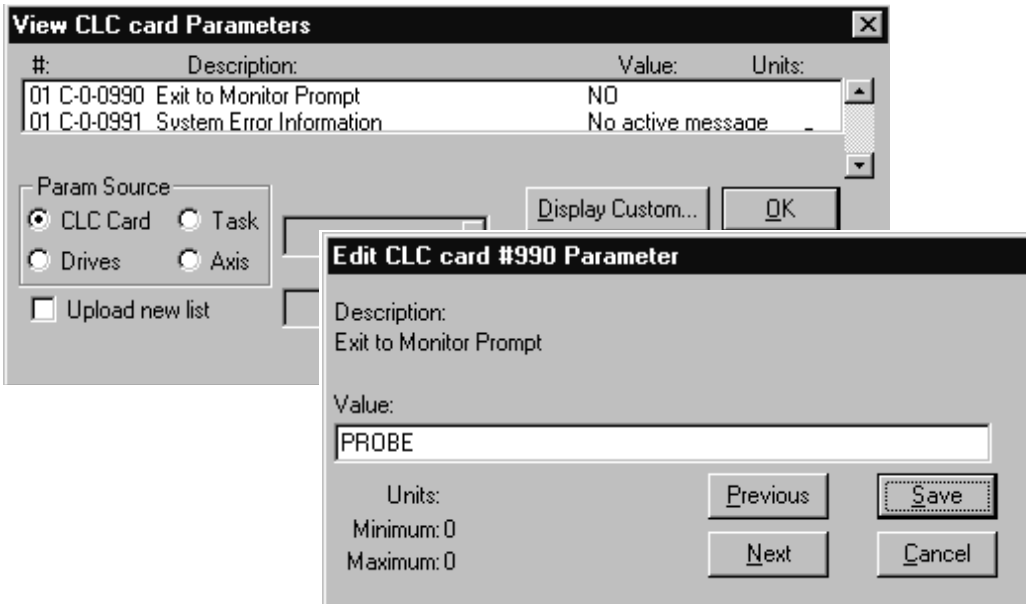
CLC-P02 Flash Firmware Upgrade

Power up the PC and system containing the CLC-P02 card.

1. Connect the IKS0061 serial cable from an available PC Com port to X27 on the CLC-P02. Switch the CLC to Parameter Mode. This is accomplished by setting register 1 bit 1 from 0 to 1, or pressing a button or switch on the machine that is mapped to register 1 bit 1.

2. Start VisualMotion Toolkit **(VMT)** and initiated communications between the DDE Server and the CLC card by selecting ***Status*** ⇒ ***System*** from the main menu. After communications have been established, close the *System Parameters* window by clicking on *Cancel.*

3. Next, select ***Setup ⇒ Overview***, and edit CLC Card parameter C-0-0990. This card parameter "*Exit to Monitor Prompt*" will contain the string "NO."



Enter the string **PROBE** in uppercase letters and Save.

1. Close VisualMotion Toolkit and the DDE server. After a few seconds, the CLC will exit to the pROBE monitor. Close any remaining pROBE Monitor windows and cancel any additional request to connect to the DDE Server.

Note: The pROBE monitor function within the CLC is comparable to exiting Windows to the DOS prompt. Only low level commands can be performed within the pROBE function.

Note: The above procedure uses VisualMotion Toolkit to activate the pROBE monitor function within the CLC card. Afterwhich, VMT is closed to free up memory resources for HyperTerminal to communicate with the CLC card via the X27 serial port.

---

5. In Windows 95/98 or NT4, use Explore to locate the HyperTerminal application icon. Double-click on the HyperTerminal icon and create a file for the PC Com port connected to X27 as follows.
   - *Bits per second*: **9600**
   - *Data bits*: **8**
   - *Stop bits*: **1**
   - *Flow control*: **Hardware**

6. Reset the CLC-P02 card using one of the following methods:
   - Recycle power to the CLC card
   - While power is applied to the card, short the two pins labeled "Reset" found next to the battery

   Afterwhich, HyperTerminal will display….

   pR0BE+>

7. To increase the communication speed of the CLC-P02, enter the following command using the pROBE function and press Enter.

   ```
   pROBE+>baud 38400
   ```

8. Now, change the baudrate of HyperTerminal to match the CLC-P02 by selecting ***File ⇒ Properties*** from the main menu and click on the ***Configure*** button.

9. In order for HyperTerminal to acknowledge the new baudrate setting, Disconnect and Reconnect HyperTerminal, press Enter and pROBE+> will once again be displayed.

10. Type the **UPDS** command followed by the Enter key to initiate firmware updating. The screen will display the following lines indicating that the CLC is now ready to accept the S-record for Flash programming.

    ```
    pROBE+>UPDS
    Updating Firmware..._
    ```

11. To begin the firmware download, select ***Transfer ⇒ Send Text File*** and navigate to the directory containing the file. Change *File types* to *ALL files (*.*),* select the firmware file (*p2gps6v56.abs*) and click on *OPEN*.

    A point "**.**" will be sent as an acknowledgment from the CLC for each line programmed. Once the download process is complete, the pROBE function will return the following message. The firmware download will take approximately 11 min 00 seconds at 38400 bps.

    ```
    Flash EPROM was successfully programmed.
    pROBE+>
    ```

12. To activate the new firmware, type **STRT** and HyperTerminal will display…

    ```
    start executive firmware? [Y/N]
    ```

    enter yes and after a few seconds, the new firmware will be running.

To confirm that the new firmware has been successfully downloaded, start VisualMotion Toolkit and select ***Status ⇒ System.*** The new firmware description will be displayed next to Version Number.

Installing and Configuring the CLC-V Motion Control Card

The CLC-V02.3 is a CLC motion control card in a VME (**V**ersa **M**odule **E**urocard) form factor.

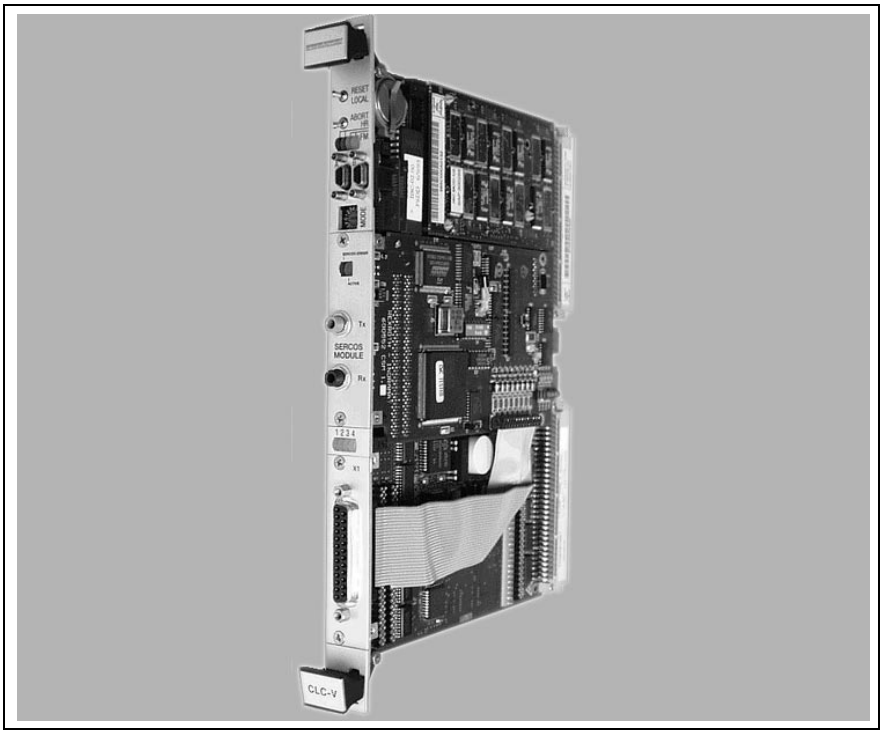


Figure 2-14: CLC-V02.3 Hardware

Caution

Electro-static discharge
Sensitive electronic device
⇒ The CLC card is a sensitive electronic device. Use caution when handling this board. Do not expose to Electro-static discharge or place the board directly on a conductive surface. Only handle the board by its face-plate or card edges.

Warning

Unexpected Movement of Machine Load
To reduce any possibility of unexpected movement during and after installation, remove all motor connects to machine load before proceeding.

⇒ Machine movement after installation should only be performed by trained personnel who understand the machine's specifications.

1. Set the CLC-V's VME bus unit number by selecting a unique number on the ***Mode Select*** rotary switch.

Note: No two VME cards can share the same unit number within the same VME rack. Every card **must** be assigned its own unique number to avoid communication errors.

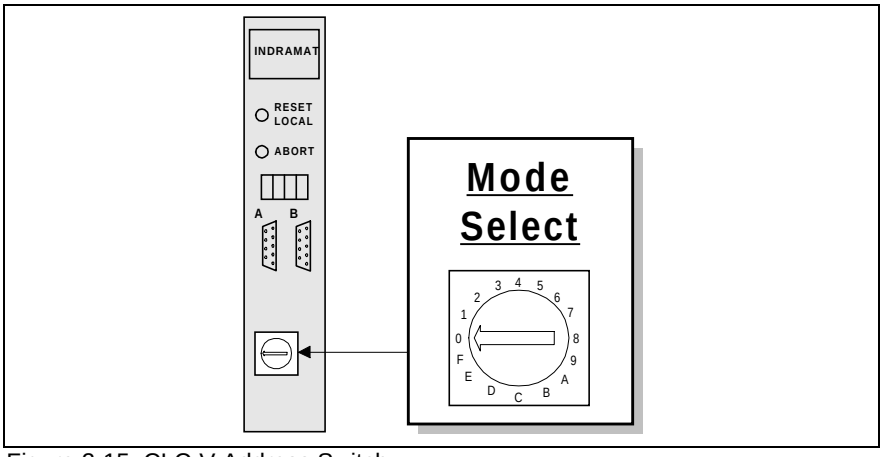


Figure 2-15: CLC-V Address Switch

2. The CLC-V card is pre-configured as the VME system arbiter or controller. However, if it is not to be used as the arbiter, disable VME Slot 1 functions by changing the position of SW5-8 on the CLC card to OFF.

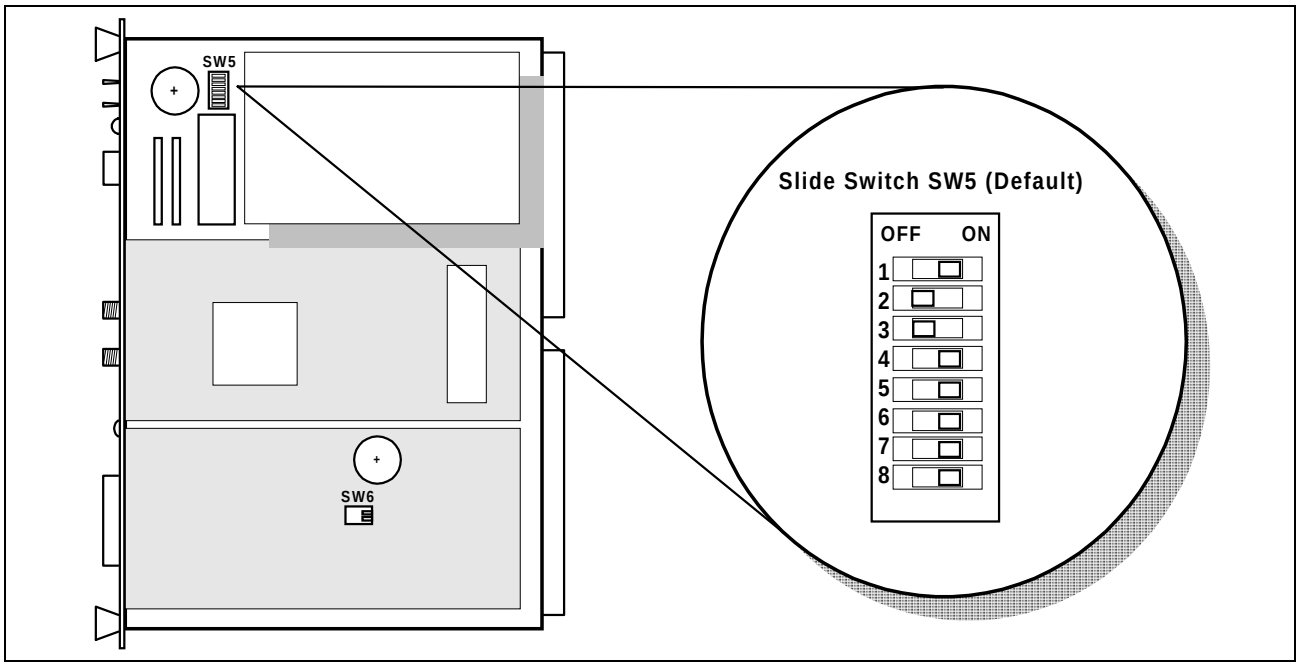


Figure 2-16: CLC-V Slide Switch Location

VME Bus request level

3. The CLC-V card is pre-configured for the highest VME bus request hardware priority, BR3. If the bus request level must be altered, change the position of switches SW5-6 and SW5-7 on the CLC card.

Switch	BR0	BR1	BR2	BR3
SW5-6	OFF	OFF	ON	ON
SW5-7	OFF	ON	OFF	ON

Table 2-5: VME Bus request level setting

4. Install the CLC-V card in the appropriate slot (slot 1 if it is the arbiter) of the VME card cage.

Caution

Non-Indramat card cages

RAM memory can be loss if VME bus does not contain a STBY line supplying battery backup.

⇒ Make sure non-Indramat card cages have a VME STBY line for battery backup.

A STBY line is simply an external power source (battery) that supplies backup power to the VME bus to maintain SRAM memory.

5. Connect the cables as shown in *Figure 2-18: CLC-V Installation* and then power up both the VME card cage and the digital drive system.

Note: Proper connection of the CLC-V card is confirmed when the red fiber optic light is transmitted through ***Tx***.

Warning

Fiber Optic Transmission Light

Bodily injuries

⇒ Do not expose your eyes to the red fiber optic transmission light. Confirm presence of light by reflecting it off your hand.

6. If VisualMotion Toolkit **(VMT)** is installed on your Laptop, continue to step 7. If VisualMotion is not installed, refer to section ***2.3, VisualMotion Toolkit Installation*** and Setup before proceeding.

7. Using cable IKS0110, make a serial connection between your laptop's Com port and serial port A on the CLC-V.

8. Start VMT and choose ***Card Selection*** from the ***Setup*** menu. Select the appropriate connection method and set the card number to match the Mode Select of step 1.

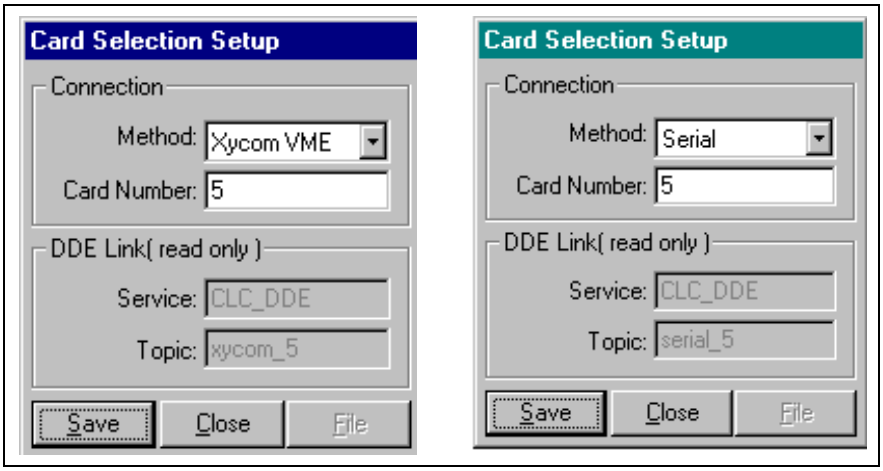


Figure 2-17: Card Selection Setup

Note: Proper VMT communication depends on the connection method selected.

- Xycom VME Bus - communication between CLC-V and Xycom PC using a VME bus.
- Serial - serial communication between PC and CLC-V using serial port A.

9. To verify VMT communications with the CLC-V card, request a system status from menu selection ***Status ⇒ System***. If communications have been established, the *System Parameters* screen will appear containing information on the installed CLC card.

For a complete description of the SW5 switches refer to ***Appendix A.4 CLC-V Hardware***

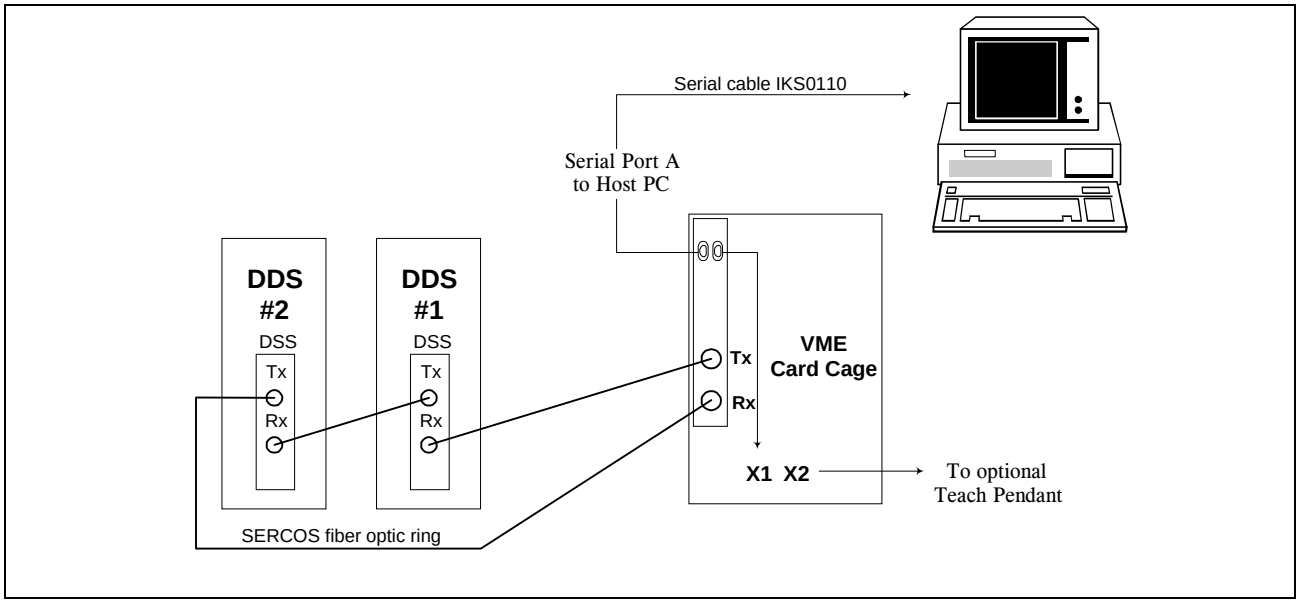


Figure 2-18: CLC-V Installation

Upgrading CLC-V flash firmware

If the current version of the CLC resident firmware is not correct or requires upgrading, a new version must be downloaded from the Host system and programmed into the non-volatile flash EPROM.

Perform a system archive to save all VisualMotion user programs and parameters before upgrading firmware.

Archiving VisualMotion System

Start VisualMotion Toolkit and select ***Archive*** from the *File* menu. Archive all programs and options, such as, cam tables, events, I/O mapper, etc, that pertain to your specific application.

Warning

Loss of VisualMotion System

User programs will be loss when a firmware upgrade is performed.

⇒ The following is a list of items that may be loss during a firmware upgrade.

- CLC System parameters C-0-xxxx
- Axis parameters A-0-xxxx
- CLC Cam tables
- Events
- I/O Mapper; FieldBus Mapper
- Points tables, Variables
- Zones
- Downloaded VisualMotion programs

Serial Port Upgrade

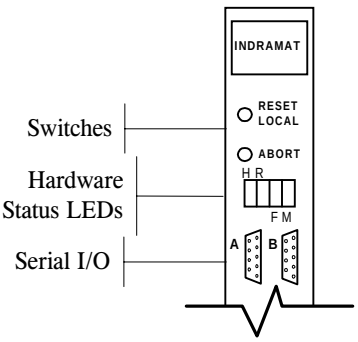
Upgrading CLC-V flash firmware using serial Port A requires the following:

- A PC or Laptop running Windows 95/98 or NT4 with HyperTerminal installed.
- An IKS0110 RS-232 serial cable for connecting from the PC Com port to serial port A on CLC-V card.
- A firmware upgrade file named according to the hardware platform and firmware version. For example, *vgp05v48.abs*

CLC-V Flash Firmware Upgrade

1. Connect the IKS0110 serial cable from the laptop or PC to serial port A on the CLC-V and power up system. Close any running instances of VisualMotion Toolkit and the CLC DDE Server, even if the programs are minimized.

2. In Windows 95/98 or NT4, use Explore to locate the HyperTerminal application icon. Double-click on the HyperTerminal icon and create a file for the PC Com port connected to serial port A as follows. Select ***Communications*** from the ***Setting*** menu and setup communication for the following:
   - *Bits per second*: **19200**
   - *Data bits*: **8**
   - *Stop bits*: **1**
   - *Flow control*: **None**



3. Switch the CLC-V to the pROBE function using the following method:
   - Place the "Abort" switch in the down position
   - Reset the CLC-V by momentarily switching the "Reset" switch upwards

   The Windows Terminal should display the following prompt from the CLC:

   ```
   pROBE+>
   ```

4. Type the **DL** download command followed by the Enter key to initiate the firmware download procedure.

   ```
   pROBE+>DL
   ```

5. To begin the firmware download, select ***Transfer ⇒ Send Text File*** and navigate to the directory containing the file. Change *File types* to *ALL files (*.*),* select the firmware file (*vgp05v48.abs*) and click on *OPEN*.

   A point "**.**" will be sent as an acknowledgment from the CLC for each line programmed. Once the download process is complete, the pROBE function will return the following message.

   The firmware download will take approximately 20 min at 19200 bps.

   ```
   xxxxx records read
   pROBE+>
   ```

   (xxxxx represents number of records transfer to CLC-V)

6. Type the **BURN** command followed by the Enter key to program the firmware file from RAM to Flash.

   ```
   pROBE+> BURN
   ```

7. At the prompt for Yes or No, type Y for Yes. The Flash programming will take between three and five minutes.

   ```
   CLC Version CLC*V*-GPS-05V48 to Flash? [Y/N] y
   Programming Flash EPROM . . .
   ```

8. When Flash Programming is done, there will be an acknowledgment if it is successful, and an error message if not.

   ```
   Flash EPROM was successfully programmed.
   CLC Version: GPS-05.48
   Start new firmware(Y=yes, N=no)?  y
   ```

   If there are no errors and the displayed version number is correct, type Y to start the new firmware.

9. Return the "Abort" switch to the middle position and reset the CLC by momentarily pushing the "Reset" switch upwards. The CLC firmware upgrade is now complete. After a few seconds, LED 1 on the front panel should be blinking, indicating that the executive program is running.

Saving CLC-V User Programs and Parameters to Flash Memory

Note: The number of erase/program cycles on Flash EPROMs is limited to 10,000. To eliminate the possibility of damage to the Flash, a SAVE command should be executed only before removing the card or when backup is needed.

Since the user programs on the CLC-V are battery-backed only through the VME standby line, they are lost when the card is removed from the VME rack. Parameters and the I-O Mapper are backed up by the primary battery on the CLC card, but this battery may also fail, making it necessary to remove the board.

All user programs, program data, CLC parameters, and the I-O Mapper may be saved in Flash EPROM before the card is removed from the rack. If the data stored in RAM is invalid upon power-up, the CLC automatically restores the data saved in Flash.

The SAVE command is entered from the pROBE+ monitor. Type **SAVE** followed by the Enter key to save data to the Flash EPROM. At the prompt for Yes or No, type Y for Yes. The CLC prints which data is being saved, and responds with an error message or acknowledgment.

```
pROBE+> save
Save user programs and parameters to Flash
(Y=yes, N=no) ?y
Saving user programs...
Saving parameters and I-O Mapper...
Flash EPROM was successfully programmed.
pROBE+>
```

The card may now be removed from the rack or the battery may be replaced without loss of program or parameter data.

Restoring CLC-V User Programs and Parameters from Flash Memory

If programs and parameters in RAM are invalid, the CLC automatically loads valid data from Flash EPROM. In some cases, it may be necessary to restore backed up data from Flash when a system is already running. The RESTore command in the pROBE+ monitor replaces the parameters and programs stored in RAM with those stored in Flash.

Type **REST** followed by the Enter key to restore data from the Flash EPROM. At the prompt for Yes or No, type Y for Yes. The CLC responds with an error message or acknowledgment.

```
pROBE+>rest
Restore user programs and parameters from Flash
(Y=yes, N=no) ?y
Data successfully restored.
pROBE+>
```

Activate the reset switch on the card or type **STRT** followed by the Enter key to restart the executive firmware.

Erasing CLC-V User Programs and Parameters from Flash Memory

It is sometimes necessary to erase all programs from RAM and Flash. Use the erase command, then remove the card from the rack or disconnect the battery. To erase data from Flash, type **eras** followed by the Enter key at the pROBE+ prompt. At the prompt for Yes or No, type Y for Yes. The CLC responds with an error message or acknowledgment. This command does not erase the executive program.

```
pROBE+>eras
Erase parameters and user programs from Flash
(Y=yes, N=no) ?y
Erasing user programs...
Erasing parameters and I-O Mapper...
Flash EPROM was successfully erased.
pROBE+>
```

Note: The number of erase/program cycles on Flash EPROMs is limited to 10,000. To eliminate the possibility of damage to the Flash, an ERASE command should be executed only when all data needs to be cleared from memory.

2.3 VisualMotion Toolkit Installation and Setup

VisualMotion Toolkit **(VMT)** is composed of 6 diskettes. VMT can be installed with single or dual language support in English and/or German. A complete Help System in both English and German are also part of the installation and contain detailed information on the use of VisualMotion along with diagnostics as content sensitive help.

Installing VisualMotion Toolkit 6.0

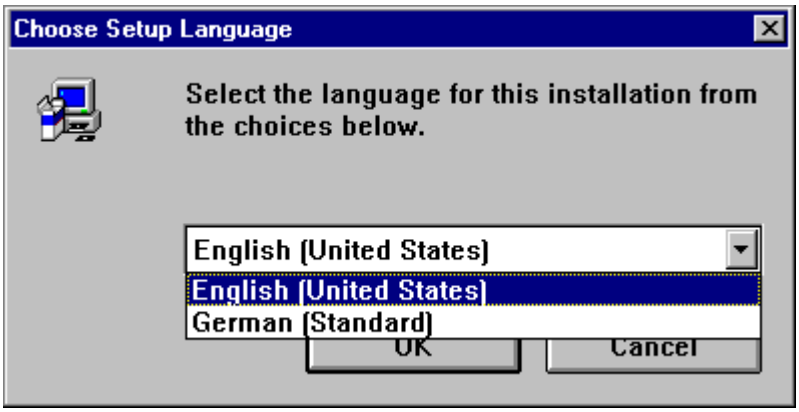
To install VisualMotion 6.0 in a host computer running under Windows 95/98 or Windows NT4 do the following:

1. Insert Disk 1 of the VisualMotion 6.0 disk set into your floppy drive.
2. Select *Run* from the *Start* menu.
3. Type *a:setup* in the Command Line box and click "OK ."

The install program will prompt you to select the language you wish to use. This option can be changed at any time after installation.

Note: To change the language, select ***Setup ⇒ Configuration*** from VisualMotion Toolkit's main menu.

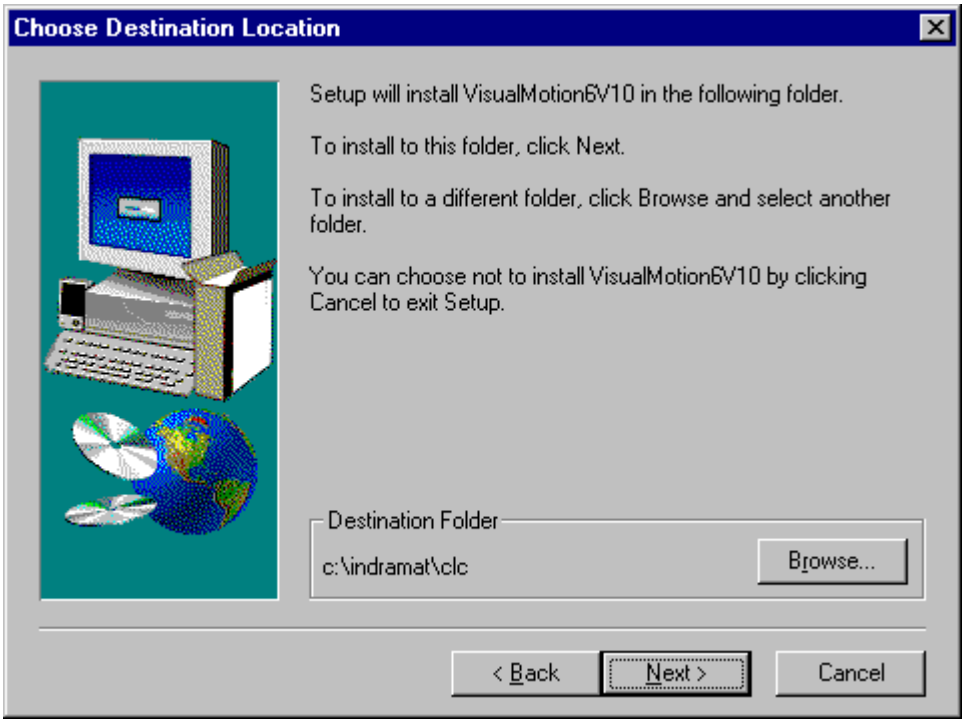
4. Select the language from the scroll list and click "OK."



An Install Shield program launches to guide you through the setup process.

5. Follow the instructions in the setup program.

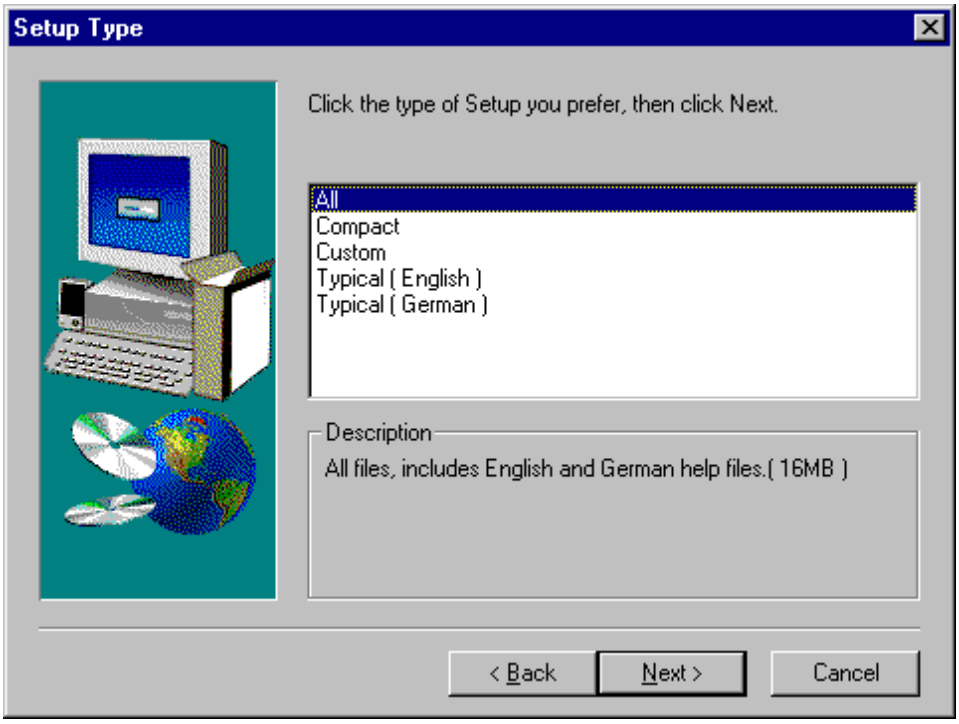
6. When the *Choose Destination Location* screen appears, you can choose where to install the VisualMotion program on your system. The default directory is *c:\indramat\clc*.


Choose Destination Location
Setup will install VisualMotion6V10 in the following folder.
To install to this folder, click Next.
To install to a different folder, click Browse and select another folder.
You can choose not to install VisualMotion6V10 by clicking Cancel to exit Setup.
Destination Folder
c:\indramat\clc
Browse...
< Back
Next >
Cancel

7. The *Setup Type* screen allows you to choose from 4 different installation types. The amount of available harddisk space required is dependant upon the setup type selected. The following table outlines how much harddisk space is required per setup type.

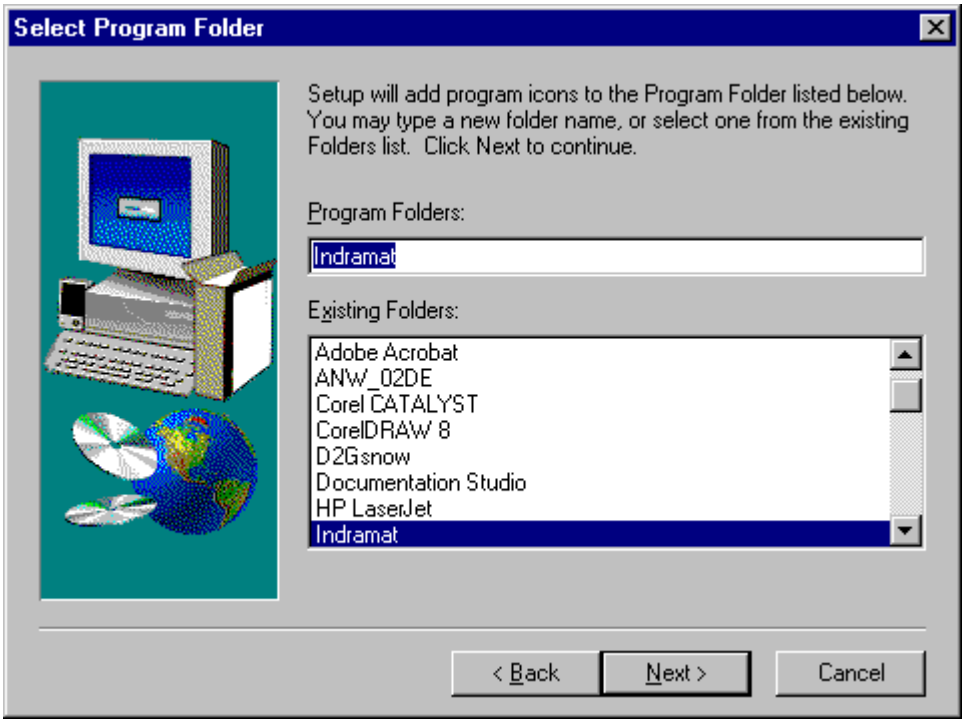
Type of Setup	Description	Required Harddisk Space
All	All files, including English and German help files	16 MB
Compact	Required files, no help files	4.5 MB
Custom	User-selectable installation	depends on selections 4.5 MB - 16 MB
Typical (English)	Required files and English help files	8 MB
Typical (German)	Required files and German help files	13 MB

Table 2-6: Setup Types



Note: The selected Setup Type also determines the number of floppy diskettes required to make a complete installation. For example, *Compact* only requires the first 4 diskettes.

8. Select the location where you would like to add program icons.



9. The setup program will show the progress of the installation and you will be prompted to insert Diskettes 2 through 6, one at a time. Insert each disk into drive A: when it is requested and click "OK."

10. The CLC Card Setup screens prompts you to select the CLC hardware card and settings to be used. The selection of a particular CLC card can be modified or changed once VisualMotion Toolkit has been completely installed.

Note: Changes to hardware cards and settings can be made within VisualMotion Toolkit under menu selection ***Setup ⇒ Card Selection.***

11. The remaining installation screens vary based on the type of CLC hardware selected. The following figures will illustrate these screens along with any additional information necessary to complete the installation.

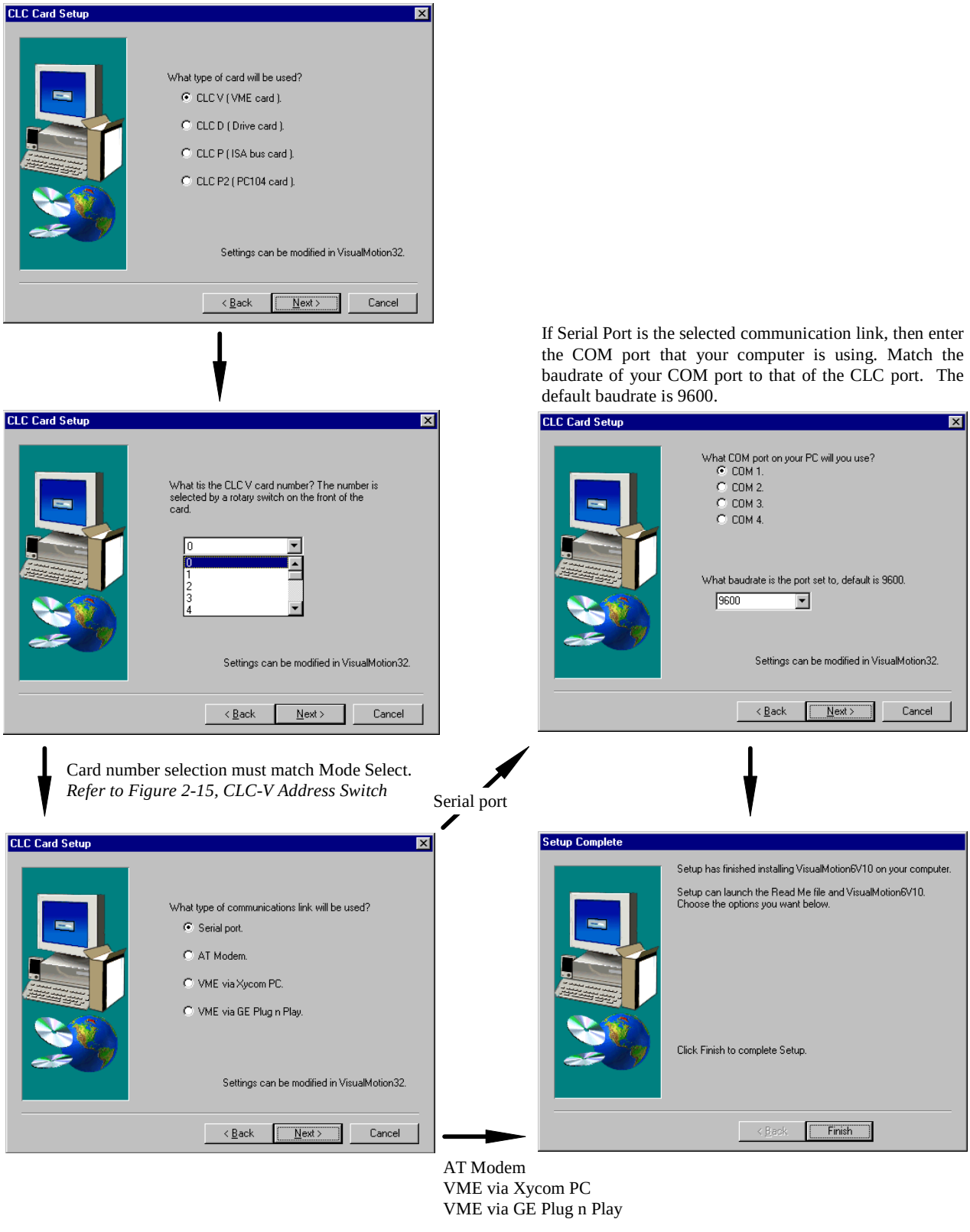
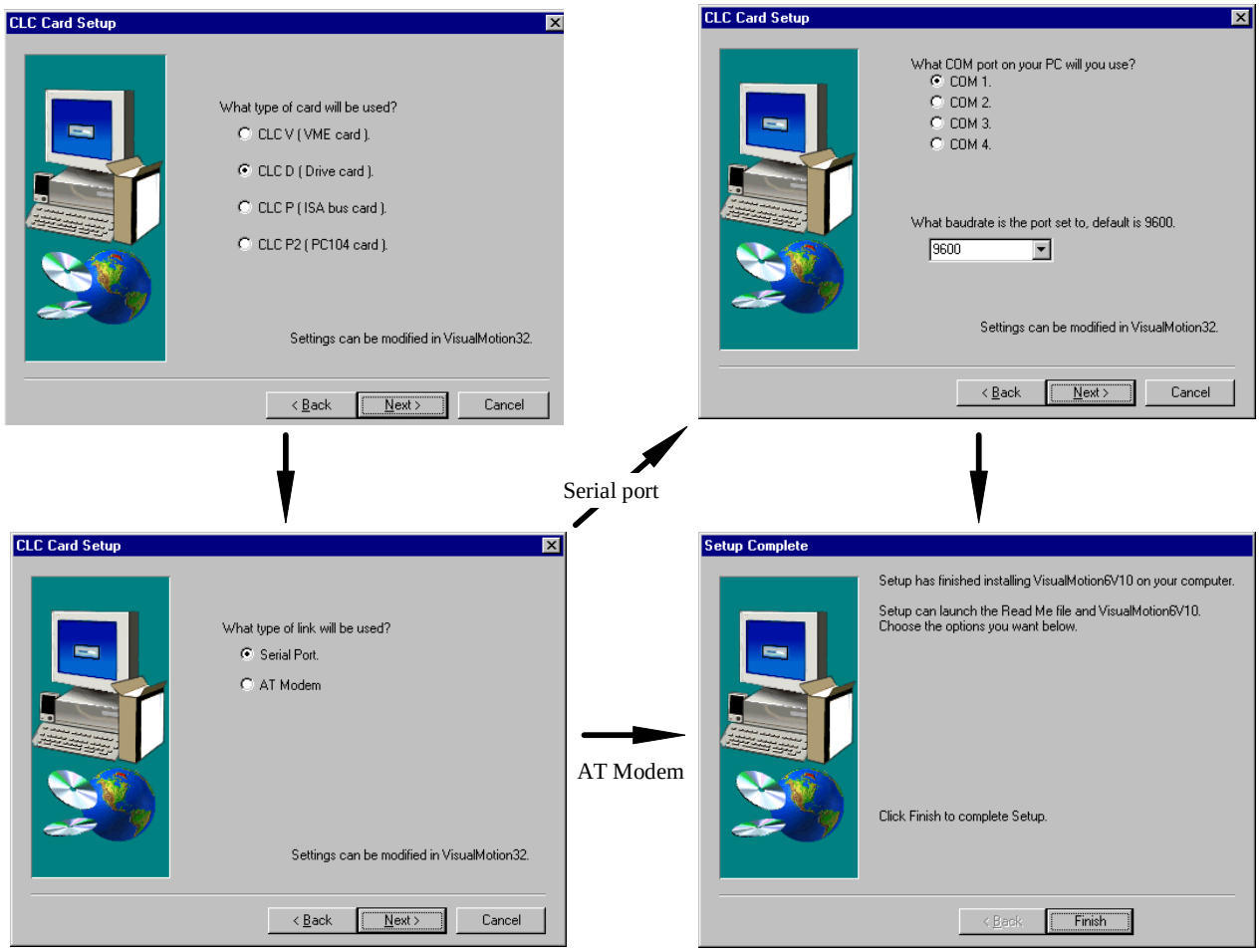
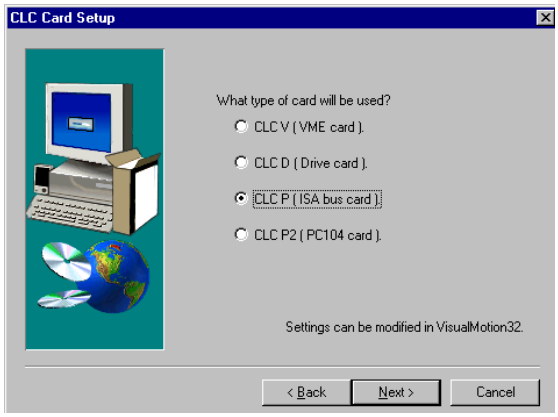

CLC Card Setup
What type of card will be used?
CLC V (VME card).
CLC D (Drive card).
CLC P (ISA bus card).
CLC P2 (PC104 card).
Settings can be modified in VisualMotion32.
< Back
Next >
Cancel
If Serial Port is the selected communication link, then enter the COM port that your computer is using. Match the baudrate of your COM port to that of the CLC port. The default baudrate is 9600.
CLC Card Setup
What tis the CLC V card number? The number is selected by a rotary switch on the front of the card.
0
0
1
2
3
4
Settings can be modified in VisualMotion32.
< Back
Next >
Cancel
CLC Card Setup
What COM port on your PC will you use?
COM 1.
COM 2.
COM 3.
COM 4.
What baudrate is the port set to, default is 9600.
9600
Settings can be modified in VisualMotion32.
< Back
Next >
Cancel
Card number selection must match Mode Select.
Refer to Figure 2-15, CLC-V Address Switch
Serial port
CLC Card Setup
What type of communications link will be used?
Serial port.
AT Modem.
VME via Xycom PC.
VME via GE Plug n Play.
Settings can be modified in VisualMotion32.
< Back
Next >
Cancel
Setup Complete
Setup has finished installing VisualMotion6V10 on your computer.
Setup can launch the Read Me file and VisualMotion6V10. Choose the options you want below.
Click Finish to complete Setup.
< Back
Finish
AT Modem
VME via Xycom PC
VME via GE Plug n Play

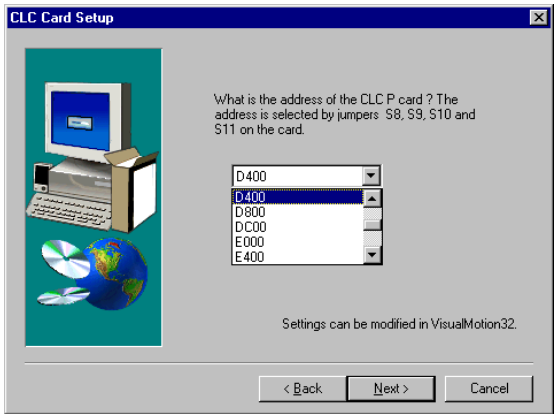
Figure 2-19: CLC-V Installation Screens

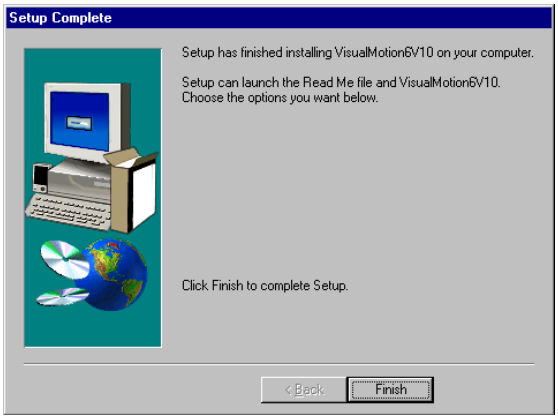

CLC Card Setup
What type of card will be used?
CLC V (VME card).
CLC D (Drive card).
CLC P (ISA bus card).
CLC P2 (PC104 card).
Settings can be modified in VisualMotion32.
< Back
Next >
Cancel
CLC Card Setup
What COM port on your PC will you use?
COM 1.
COM 2.
COM 3.
COM 4.
What baudrate is the port set to, default is 9600.
9600
Settings can be modified in VisualMotion32.
< Back
Next >
Cancel
Serial port
CLC Card Setup
What type of link will be used?
Serial Port.
AT Modem
Settings can be modified in VisualMotion32.
< Back
Next >
Cancel
AT Modem
Setup Complete
Setup has finished installing VisualMotion6V10 on your computer.
Setup can launch the Read Me file and VisualMotion6V10. Choose the options you want below.
Click Finish to complete Setup.
< Back
Finish

If Serial Port is the selected communication link, then enter the COM port that your computer is using. Match the baudrate of your COM port to that of the CLC port. The default baudrate is 9600.

Figure 2-20: CLC-D Installation Screens

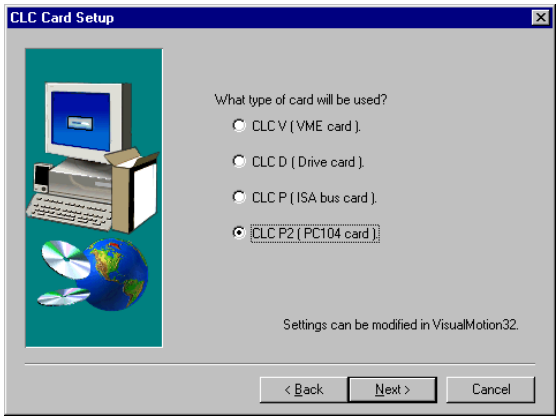

CLC Card Setup
What type of card will be used?
CLC V (VME card).
CLC D (Drive card).
CLC P (ISA bus card).
CLC P2 (PC104 card).
Settings can be modified in VisualMotion32.
< Back
Next >
Cancel

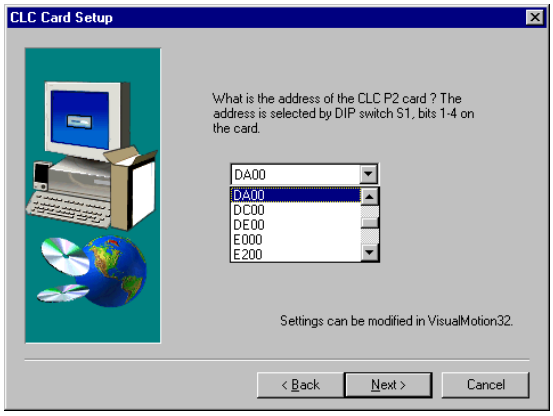

CLC Card Setup
What is the address of the CLC P card ? The address is selected by jumpers S8, S9, S10 and S11 on the card.
D400
D400
D800
DC00
E000
E400
Settings can be modified in VisualMotion32.
< Back
Next >
Cancel

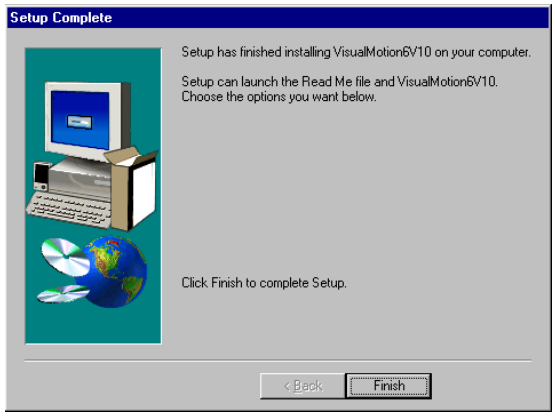

Setup Complete
Setup has finished installing VisualMotion6V10 on your computer.
Setup can launch the Read Me file and VisualMotion6V10. Choose the options you want below.
Click Finish to complete Setup.
< Back
Finish

CLC-P01 address must match base address selected when the card was installed. Refer to *Table 2-1, CLC-P01 card number and base address memory selection.*

Figure 2-21: CLC-P01 Installation Screens


CLC Card Setup
What type of card will be used?
CLC V (VME card).
CLC D (Drive card).
CLC P (ISA bus card).
CLC P2 (PC104 card)
Settings can be modified in VisualMotion32.
< Back
Next >
Cancel




Setup Complete
Setup has finished installing VisualMotion6V10 on your computer.
Setup can launch the Read Me file and VisualMotion6V10. Choose the options you want below.
Click Finish to complete Setup.
< Back
Finish

CLC-P02 address must match base address selected when the card was installed. Refer to *Table 2-2, PC/104 memory address selection.*

Figure 2-22: CLC-P02 Installation Screens

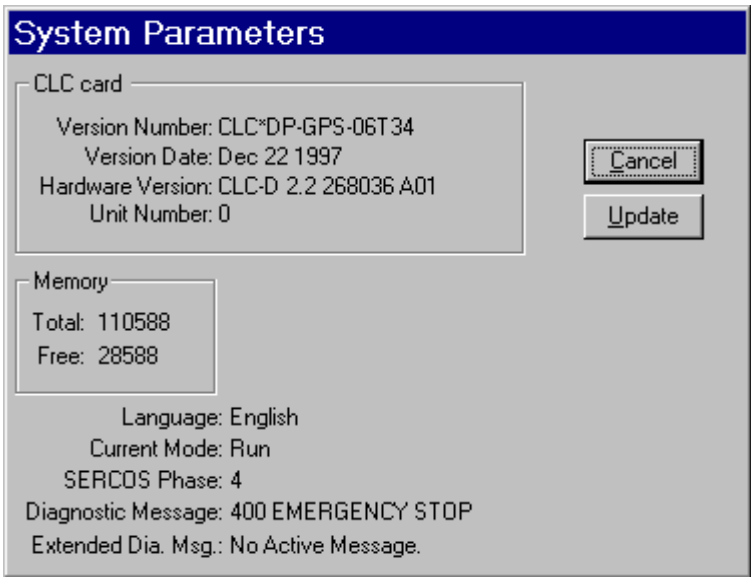
Once VisualMotion Toolkit **(VMT)** is installed, verify communications by performing the following procedure.

⇒ Start VMT by selecting ***Start ⇒ Program Files ⇒ Indramat ⇒ VisualMotion***

⇒ Verify that the correct CLC hardware and card number are set within ***Setup ⇒ Card Selection***

⇒ Select ***Status ⇒ System*** from the main menu

If proper communication has been established, the System Parameter screen (shown below) will display the CLC version number along with other messages.



3 Create/Download Sample Program

3.1 Introduction

This chapter presents instructions on how to create and download a sample program. It is a basic single axis program that is not application specific. Follow the instructions outlined below. ***Refer to the VisualMotion 6.0 Reference Manual for detailed icon descriptions.***

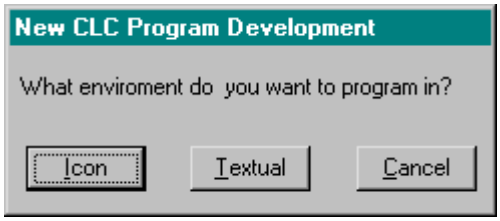
3.2 Sample Program

1. Start VisualMotion Toolkit (**VMT**) by either double-clicking on a VMT icon shortcut or selecting …

   Start ⇒ Programs ⇒ Indramat ⇒ VisualMotion6Vxx.

   xx = current revision level of VisualMotion Toolkit

2. From VMT's main menu select ***File ⇒ New*** and choose ***Icon*** as your programming environment.



3. The following series of programming icons will be used in this example. As the icons are placed on the screen, dialog boxes will open and prompt you to enter setup information. A description of the icons along with the setup information needed for each dialog box will be provided.

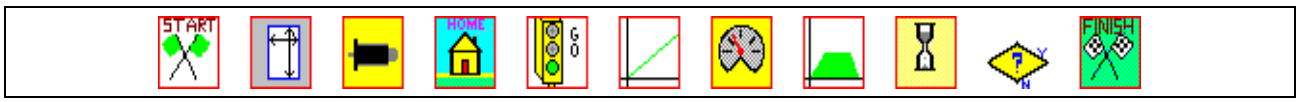


Figure 3-1: Sample Program Icons

Note: The toolbar programming icons found above the programming area are visible regardless of the Icon palette selected from the *Options* menu.

Toolbar Programming Icons

Start Icon

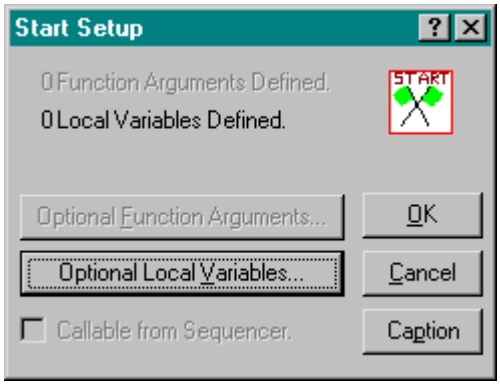
4. Select and place the ***Start*** icon from the set of toolbar icons onto the VisualMotion programming area. All VisualMotion tasks and subroutines must begin with a ***Start*** icon.

Note: Once an icon is selected, the cursor will change to a crosshair. Move the crosshair onto the programming area and click to place. To make best use of the programming area, begin placing icons in the upper most left-hand corner.

Task: A task is simply a process used by the programmer to perform a machine operation by using VisualMotion programming icons.

Subroutine: Subroutines are basically sub-programs that are programmed to start within the main task.

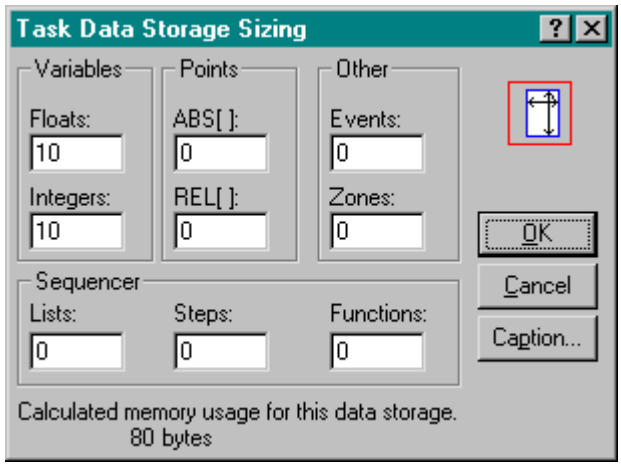
Since this program does not contain any optional function arguments or local variables select *OK*.



Size Icon

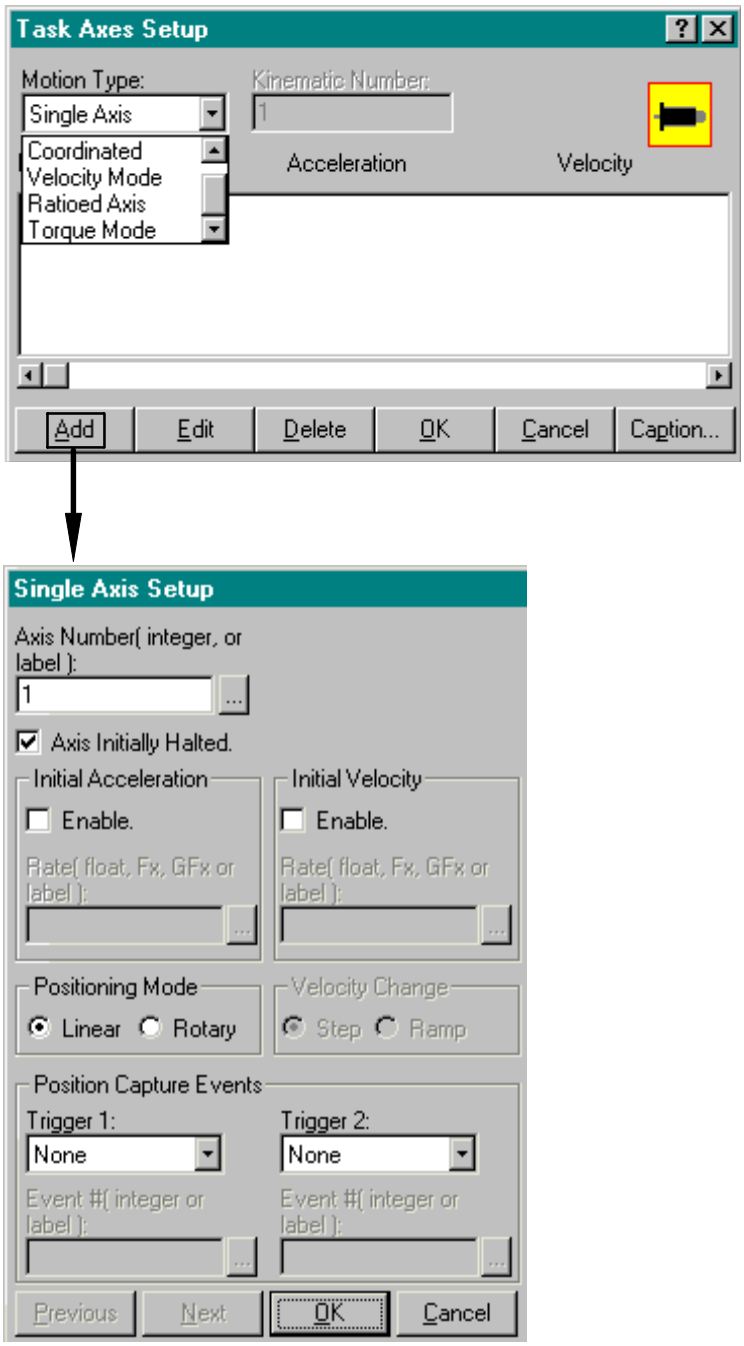
5. Select and place the ***Size*** icon from the 'Single' Icon Palette to the right of the *Start* icon. The ***Size*** icon determines the number of variables, points, events and zones to be used in the VisualMotion program. It also limits the number of Sequencer Lists, Steps and Functions. *Refer to the VisualMotion 6.0 Reference Manual for more information.* To maximize memory on the CLC, place limits on the data storage space.

Note: The default size for both Floats and Integers is 50. Change this value to 10 for each and click on *OK*.



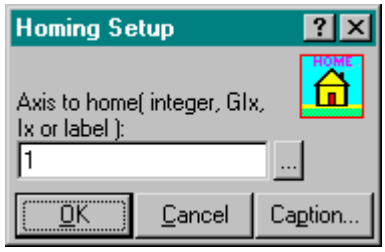
Axis Icon

6. Select and place the ***Axis*** icon to the right of the *Size* icon. The ***Axis*** icon configures the primary operation mode for the axes to be used in this Task.
   a. Choose "***Single Axis***" from the Motion Type drop down menu box.
   b. Click on the ***Add*** button to setup the axis.
   c. Configure the axis according to the Single Axis Setup window shown below. When done, select ***OK*** and then ***Cancel*** to close the Single Axis Setup window. Select **OK** once again to close the Task Axes Setup window.



Home Icon

7. Select and place the ***Home*** icon to the right of the *Axis* icon. The Home icon executes the drive-controlled homing procedure.



Enter a **1** and then select *OK*.

Note: Before and axis can be Homed using the ***Home*** icon, a homing routine must be setup. Refer to the following procedure for details.

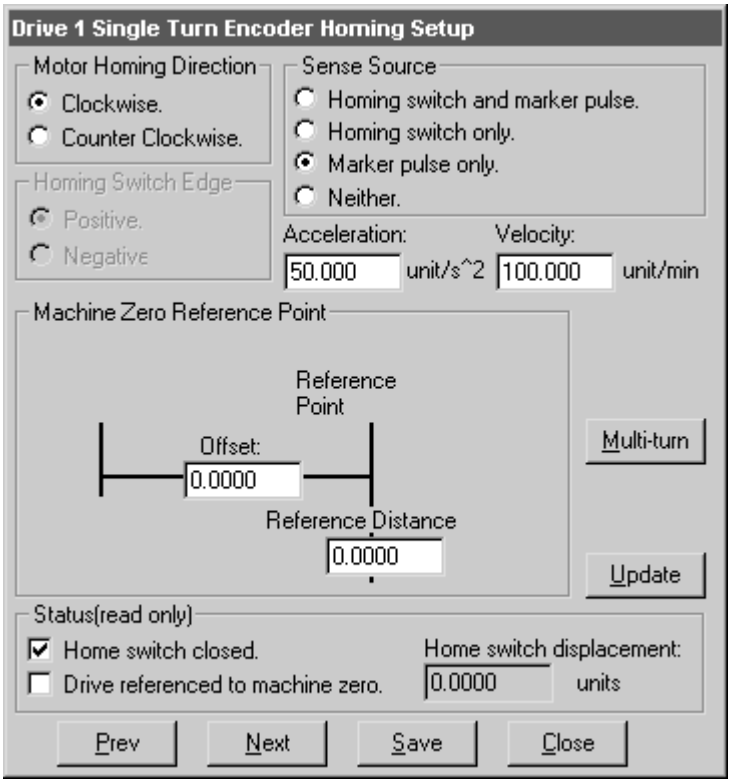
Homing Procedure

Select ***Setup*** ⇒ ***Drives*** from VMT's main menu to open the CLC Drive Parameter Editor window. Now, select ***Parameters*** ⇒ ***Drive Reference*** and VMT will automatically sense the active drive's motor encoder type and launch either the single or multi-turn encoder setup window.

Note: Verify that the motor and drive are properly connected and powered up. Check the serial communication cable between the CLC card and host computer for proper connection.

Drive 1 Single Turn Encoder Homing Setup

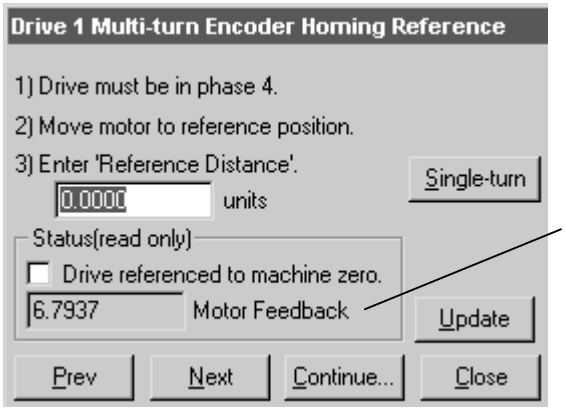
Setup the motor's homing routine according to the window below. The Acceleration and Velocity parameters should be set to a low enough value as to not cause sudden jerk movement. When done, click on *Save* and then *Close* to complete the homing routine for Drive 1.



The homing procedure is an internal function of Indramat's intelligent digital drives and requires only that VisualMotion send a ***Home*** command to the drive. The actual homing procedure performed by the drive is set by the drive's parameters.

Drive 1 Multi-turn Encoder Homing Reference

If the motor you are using contains an Absolute encoder, the following window will automatically be displayed when ***Parameters ⇒ Drive Reference*** is selected from the CLC Drive Parameter Editor window.

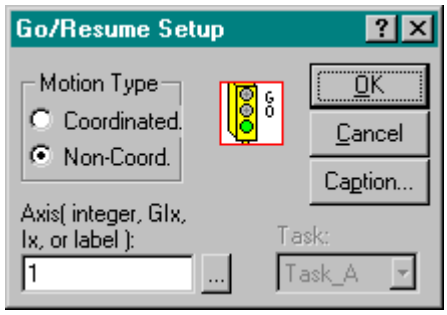


When done, click on Close to exit this window. To close the CLC Drive Parameter Editor window and return to the Task programming screen, select ***File ⇒ Exit***.

The Homing routine for the Home icon is now complete. *For more information regarding the CLC Drive Parameter Editor, refer to the VisualMotion 6.0 Reference Manual.*

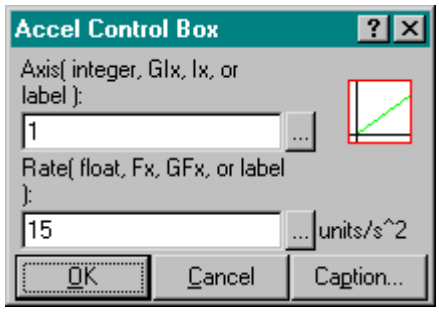
GO Icon

8. Select and place the ***GO*** icon to the right of the *Home* icon. The GO icon enable the axis' drive ready signal (RF.)
   a. Enter a **1** in the Axis field to specify which axis to initiate.
   b. Non-Coord. Motion should be selected for a single axis Motion Type.



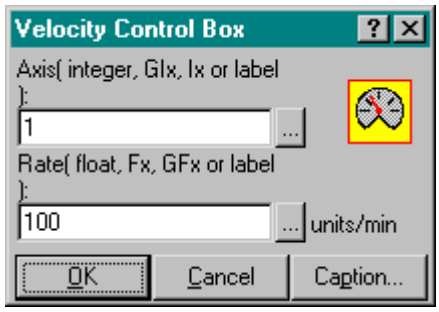
Accel Icon

9. Select and place the **Acceleration** icon to the right of the *GO* icon. The **acceleration** icon sends the acceleration rate to the drive that will be used in the move calculation.
   a. Enter a **1** to specify the axis
   b. Enter **15** for the rate of acceleration and click on *OK*.



Velocity Icon

10. Select and place the ***VELOCITY*** icon to the right of the *Accel* icon. The velocity icon sends the velocity rate to the drive that will be used in the move calculation.
    a. Enter a **1** to specify the axis
    b. Enter **100** for the velocity rate click on *OK*.

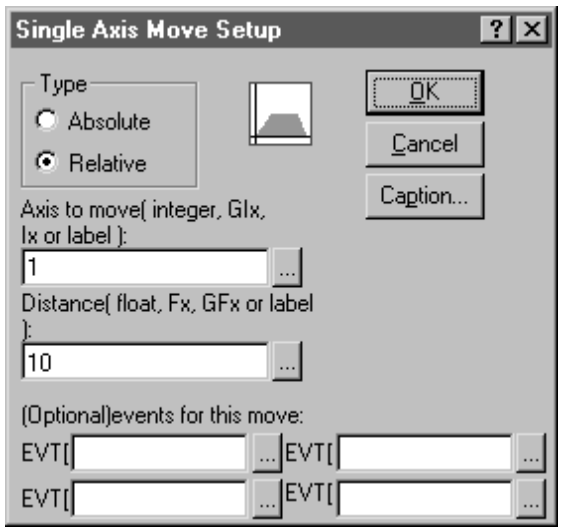


MOVE Icon

11. Select and place the **MOVE** icon to the right of the *Velocity* icon. The *velocity* icon sends the move distance to the drive and initiates the move.
    a. Select **Relative** as the move Type.

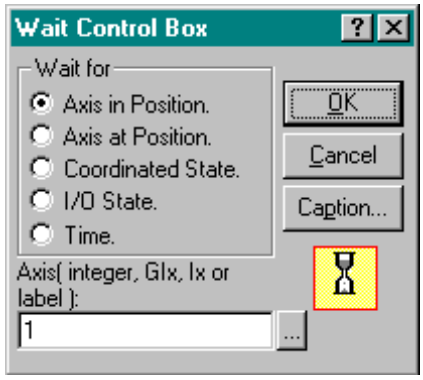
Note: A ***Relative*** move is an incremental distance that is moved every time the move icon is encountered in the program flow. An ***Absolute*** move is a exact position that is reached when the move icon is encountered and is not repeated unless the absolute position changes.

    b. Enter a **1** to specify the axis number
    c. Enter a distance of **10** and click on *OK*

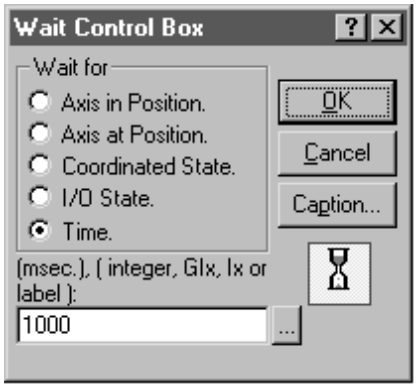


WAIT Icon

12. Select and place a **WAIT** icon to the right of the *MOVE* icon. The task execution will *wait* at this point until the *wait condition* is true. In this case, the task waits until axis 1 is in position.

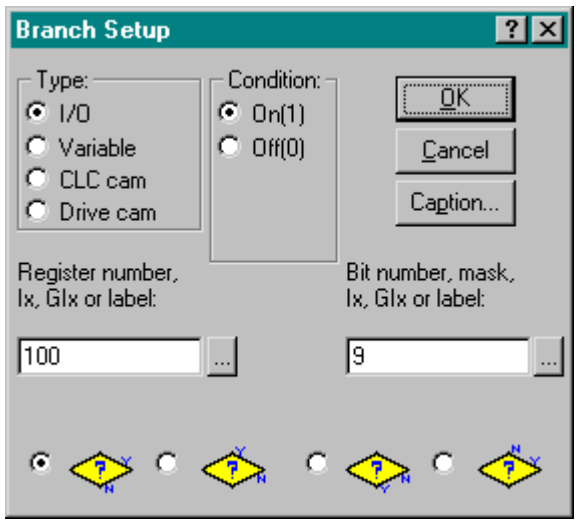


13. Add a second **WAIT** icon below the first and enter a *Time* of 1000 msec. This icon will introduce a pause of 1 sec to the program before proceeding to the next relative move.



Branch Icon

14. Select and place the **Branch** icon to the right of the *WAIT* icon. The **Branch** icon re-directs the program flow depending upon a true/false logical value. This creates a loop within the program depending on the value of Register 100.



Note: The **Branch** icon will loop back to a specified icon until the branch condition is true. In this example, the *Finish* icon will not be encountered until register 100 bit 9 is On(1.)

Finish Icon

15. Select and place the **Finish** icon to the right of the *Branch* icon. When encountered, the program will end. All *tasks* and *subroutines* must end with the Finish icon.

Line Icon

16. Use the Line icon to connect the icons and show program flow. To connect the icons, click once with the left mouse button on the first icon and then click on the next icon in the program flow. A line will join the icons with an arrow indicating program flow.

Note: If an error is made while connecting two program icons, use the

CUT icon to remove the connection line created.

Afterwards, re-select the line icon to continue.

The completed program should appear as shown below:

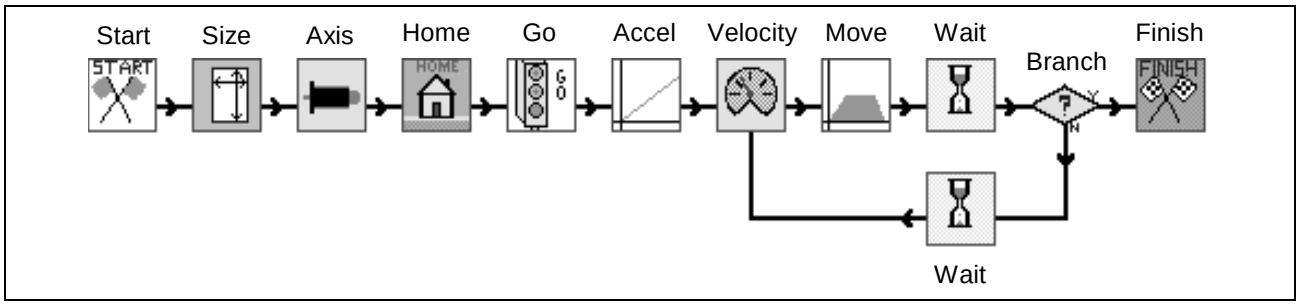


Figure 3-2: Completed sample program

Note: In order to create the loop from the **Branch** icon back to the **Velocity** icon, click on the *Branch* icon first then click on the *Wait* icon. Repeat the step starting with *Wait* icon and finish with the *Velocity* icon.

Save, Compile and Download Program

Save Program

1. To save the example program to a harddrive, select ***File*** ⇒ ***Save As*** and enter the filename "sample.str."

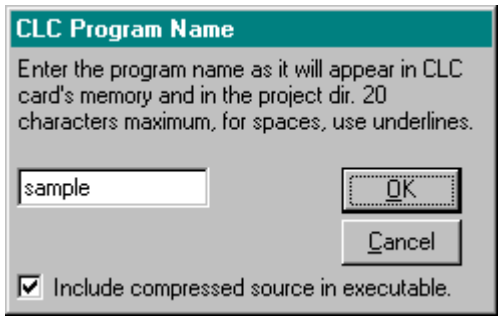
Note: Icon program files are saved with a ".str" extension at the end of the filename. **See Section 3.4 for other extensions and their descriptions.**

Note: Selecting *Save, Compile, Download* from the *File* menu, or clicking on the toolbar icon , will automatically cover steps 1-3 below.

Compile Program

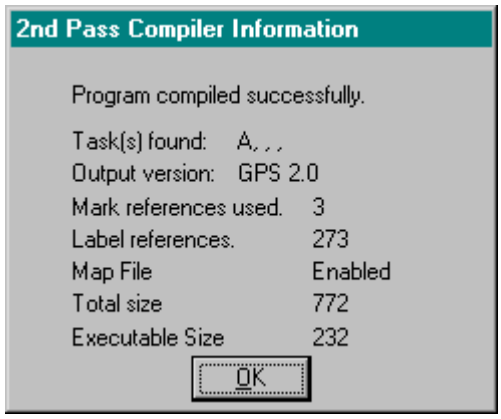
2. Compile the program by choosing ***Compile*** under the *File* menu.
   a. The compiler will first check for a complete path from "Start" to "Finish" for each Task and subroutine.
   b. The compiler will then prompt you to enter a name for the program as it will appear on the CLC card. Use the default name in the text box.

   The compiler will then convert the program to code.


CLC Program Name
Enter the program name as it will appear in CLC card's memory and in the project dir. 20 characters maximum, for spaces, use underlines.
sample
OK
Cancel
Include compressed source in executable.

Note: Checking the "Include compressed source in executable" box in the above window enables the option of compiling and downloading the program icon file (*.str) along with the *.exc file onto the CLC card. The file size will be larger; however, the icon program file can be retrieved from the executable file on the card in case of a harddrive failure.

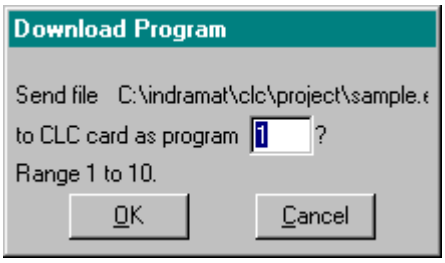
Click on *OK* and the 2nd compiler information window will appear indicating a successful compile and provide information on the compiled program.



Press *OK* to close the Compiler Information window.

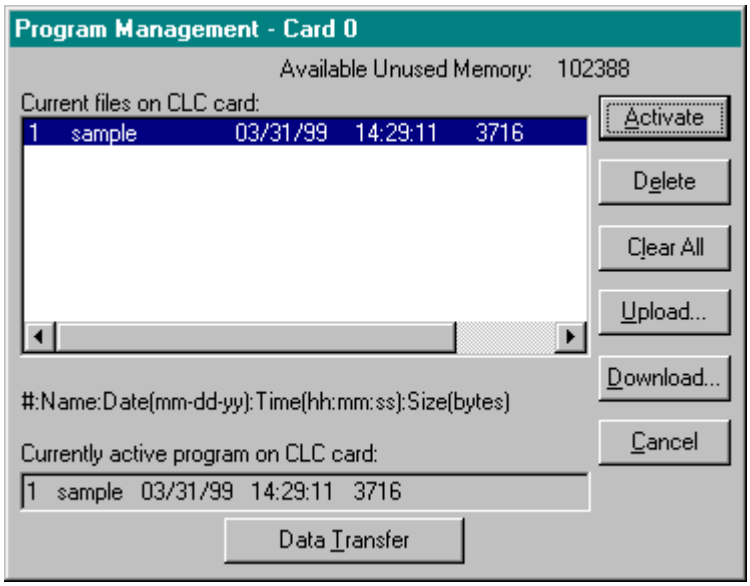
Download Program to CLC

3. Download the program to the CLC and activate it.
   a. Select ***Program Management*** under the *File* menu.
   b. Click on the "Download" button
   c. Select the program that you just created and click on *Open*. The program will be listed as "sample.exc" under the *Project* folder.
   d. In the Download Program window, enter a program number from 1 to 10 that will be used to identify the VisualMotion program when downloaded to the CLC card.



4. After the download is complete, the program will be automatically highlighted and active in the CLC card.

Note: To activate a different program on the CLC card, simply click and highlight the desired program in the *Current files* field and click on the *Activate* button.



To close the Program Management window, click on *Cancel*.

3.3 Program Variables

The CLC allows floating point variables, integer variables and constants. There are three types of floating point and integer variables, global, program and local.

Global variables, designated GF[#] (Global Float) and GI[#] (Global Integer), are stored in CLC RAM and their values are not retained during power off. There are 256 global floating points and 256 global integers and they are shared among the programs stored on the CLC card. They can also be used to exchange values between external components of a CLC system that are capable of accessing the global memory area.

Program variables are designated F[#] and I[#]. The number of program variables allocated to a CLC program is determined by the Size icon in VisualMotion. Program variables retain their values during power off. The variables can be addressed in a user program by assigning a label to the variable number.

Local or *stack based* variables exist only while in the function (task, subroutine, or event) where they are declared. Local variables are used within a subroutine for local data only. They don't exist outside the subroutine. This type of variable is useful for temporary results within a function or to pass values to a function.

Task VisualMotion can have up to 4 tasks running in each program. Tasks A-D run simultaneously and are given equal priority (task A is executed first.) A task is a process that the user runs in his machine. Using VisualMotion, the user can have 4 separate processes or task running simultaneously and each task can be independent of each other.

Subroutine Subroutines are basically sub-programs that are called by the main program when selected to start. They are used mainly to improve readability as well as simplify the program.

Event functions Events are basically interrupt driven subroutines. They can be triggered by a variety of methods, such as transition of an input, repeating timer, position trigger, etc.

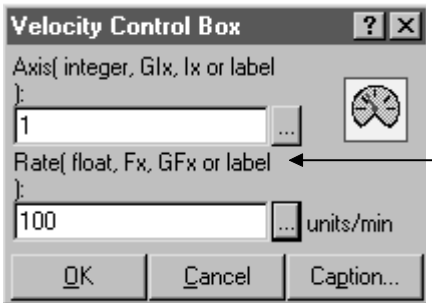
Assigning Labels to Variables

Select ***Labels ⇒ User Labels*** under the *Edit* menu to assign a label to a variable. A label is simply a name given to a variable which can help the user identify its function when programming. This can also be done directly within an icon dialog box using the following procedure:

Many programming icons contain a User label button that opens the User defined Labels window giving the programmer the ability to create program variables.

Note: Variables can be used to replace numerical values within programming icons. Numerical values within icons that are compiled and downloaded to the CLC cannot be modified unless changed and re-compiled. On the other hand, once compiled and downloaded, variables can be modified within VisualMotion Toolkit by selecting ***Data ⇒ Variables***. Modified variables are active the next time that specific icon is encountered in the program flow.

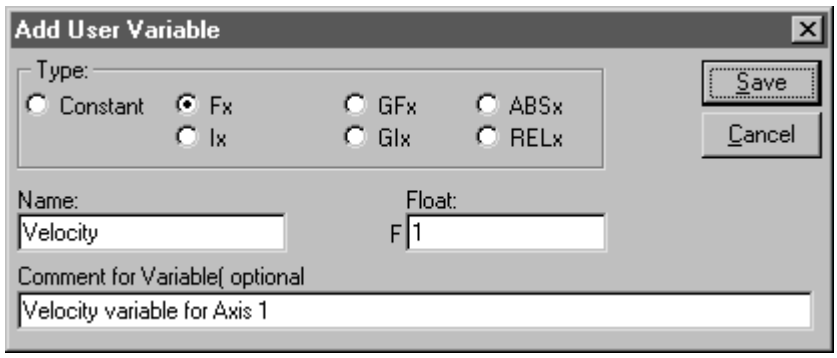
Example: Velocity Icon



Each icon displays an allowable variable type within the Rate field.

1. Click on the User label button . The ***User Defined Labels*** window will open.

2. Click the *Add* button to open the ***Add User Variable*** window.



3. Select the ***Type*** of variable to add. Example: constant, floating point or integer.

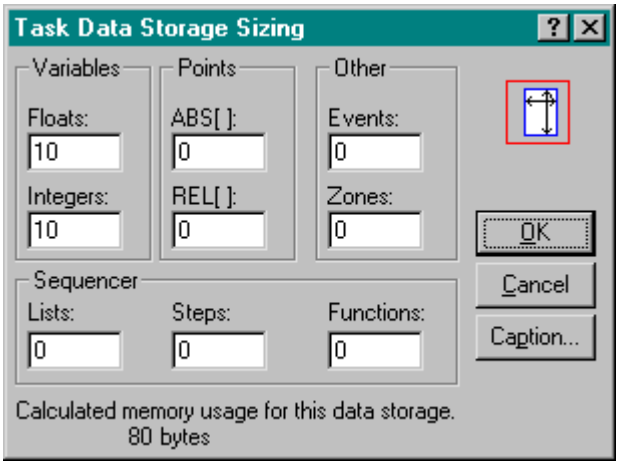
Note: Allowable variable types are listed within the icon's dialog window.

---

4. Enter a ***Name*** for the variable

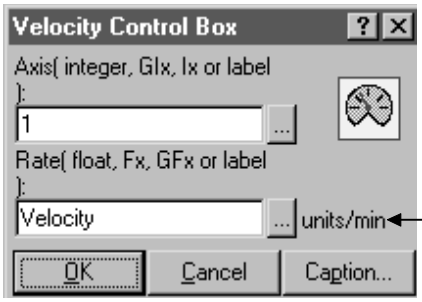
5. Assign a **1** to the Float(F) label. A second Float variable will receive a 2.

Note: The number of available *Variables* is determined by the Size icon.



6. Click *Save* to add the variable, then *Cancel* to close the *Add User Variable* window.

7. Highlight the new variable in the *User Defined Labels* window and click *OK*. This will place the new label in the Rate field of the icon's dialog window.



The velocity's rate is now a variable. Click on OK to complete the variable process.

8. Now, use the same procedure to add variables for the **MOVE** icon and the second lower **WAIT** icon.

After all the variables are assigned, Save, Compile and Download the VisualMotion program and activate. To view programmed variables within a VisualMotion program select ***Data ⇒ Variables***.

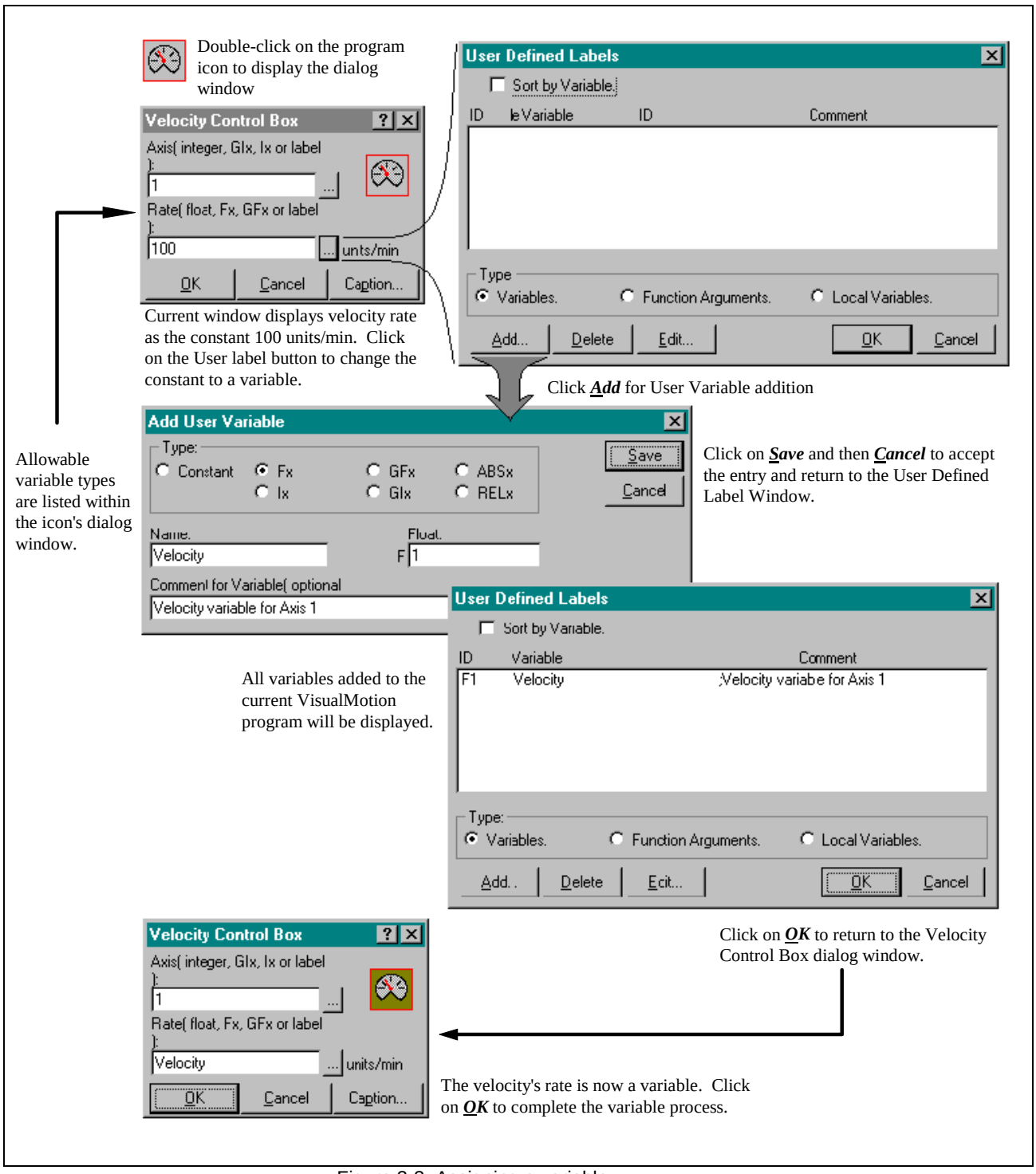


Figure 3-3: Assigning a variable

Assigning a Value to a Variable

VisualMotion variables are defined by the programmer and are used in programs to enable the user to modify a value in the active VisualMotion program without having to re-compile and download the information. Modified variables are only updated the next time the program cycles and encounters an icon instruction using that variable. Select ***Data ⇒ Variables*** to view the Active Program, Variable window.

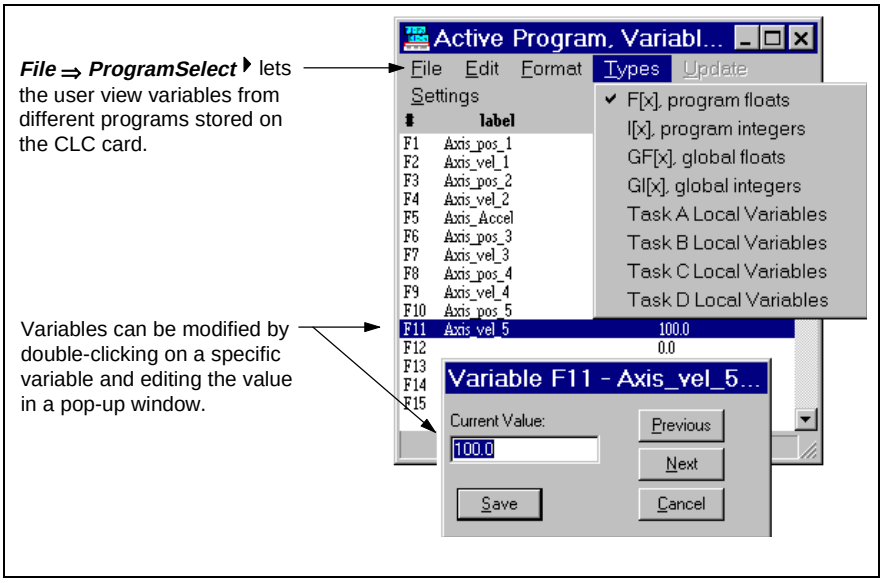


Figure 3-4: Viewing and editing variables

The following variable types are available:

Floating Point Variables (F1-Fx)

A floating point variable is simply a number containing a decimal point. The number of 32 bit floating point variables is defined in the sizing icon at the beginning of the program and are stored as part of the program.

Integer Variables (I1-Ix)

Integers are signed or unsigned whole numbers, such as 5 or -3. The number of 32 bit Integer variables is defined in the sizing icon at the beginning of the program and are stored as part of the program.

Global Floating Point and Integer Variables (GF1-Gfx;GI1-GIx)

Global variables are available to all programs stored on the CLC card. Global variables are program independent. Multiple programs can write to the same set of Global variables.

Task A-D Local Variables

Local variables are created when a subroutine or task begins and eliminated when the subroutine or task execution has ended. Arguments can be passed to local variables to allow multiple applications of a common subroutine.

3.4 File Types

Visual Motion uses a number of different file types. Refer to the following chart to identify the file type according to its extension.

Extension	Description
.acc	Text file that ACAM utility converts to a .csv file
.csv	Comma-Separated-Variable type file used to store cam profiles.
.exb	Compiled program file that is uploaded from the CLC. It is ready to run and contains program data.
.exc	Compiled program file that is downloaded to and executed by the CLC.
.iom	I/O mapper file. Text file consisting of Boolean strings.
.lss	Text file where Visual Motion stores register and bit labels used by the .str file.
.lst	Text file that is referred to for register and bit labels when the registers on the CLC are viewed.
.map	File used by the “Show Program Flow” function to trace the flow of the program while it is executing.
.pnt	Absolute Point Table
.pos	Text file that PCAM utility converts to a .csv file
.prm	Parameter file in archived format. These files can be transferred to the CLC.
.str	Graphical icon program file displayed in VisualMotion Toolkit
.tbl	Text file of points created by the CLC “Oscilloscope” function.
.var	Old variable file
.vel	Text file that PCAM utility converts to a .csv file
.vtr	New variable file
.mtn	Text language program source file.
.zon	Zone File

4 Program Execution

4.1 Introduction

This chapter covers CLC program execution which includes a brief description of the System and Task Control Registers and the I/O Mapper function. A brief parameter overview has also been included along with some troubleshooting and communication information. ***For a more detailed discussion on any of these topics refer to the VisualMotion 6.0 Reference Manual***

4.2 Initial Setup Prior to Operation

The following procedure covers the initial setup required to run the sample program that was created and downloaded in the previous chapter. The sample program should be open within VisualMotion Toolkit and Activated on the CLC card.

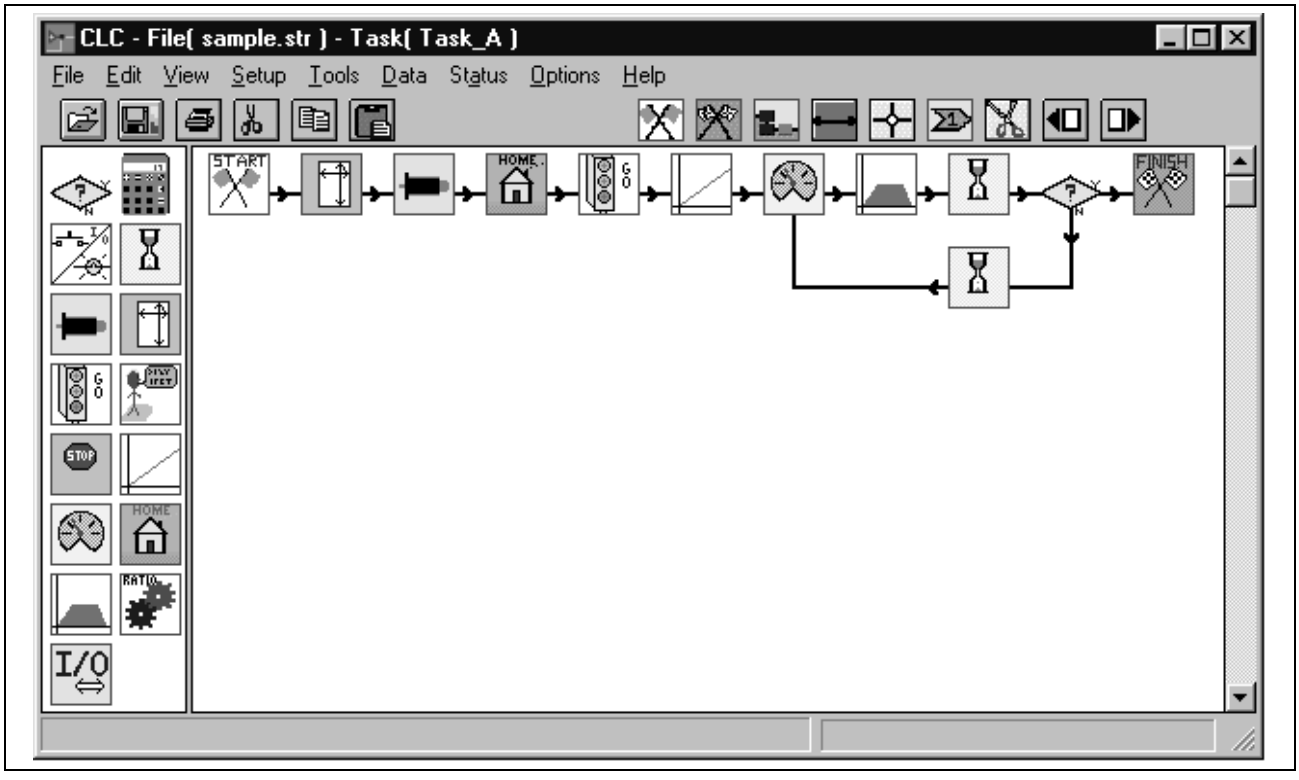
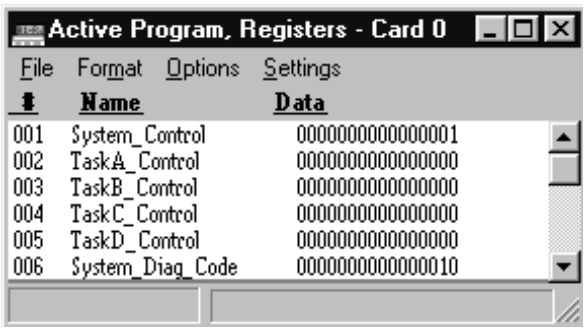


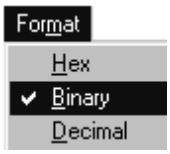
Figure 4-1: Sample VisualMotion Toolkit Program

Control Registers

Before a VisualMotion program can be activated, key register bits must be set in the System and Task Control registers. Registers 001, *System_Control* and 002, *TaskA_Control* are dedicated as system registers and are used to control program operation. Although these registers an be modified by the user, we will create an I/O Map using the I/O Mapper function to associate register 100, *User_Inputs_Reg1*, with registers 001 through 005.

VisualMotion registers can be displayed by selecting ***Data ⇒ Registers*** from VisualMotion Toolkit's (**VMT**) main menu.

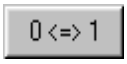




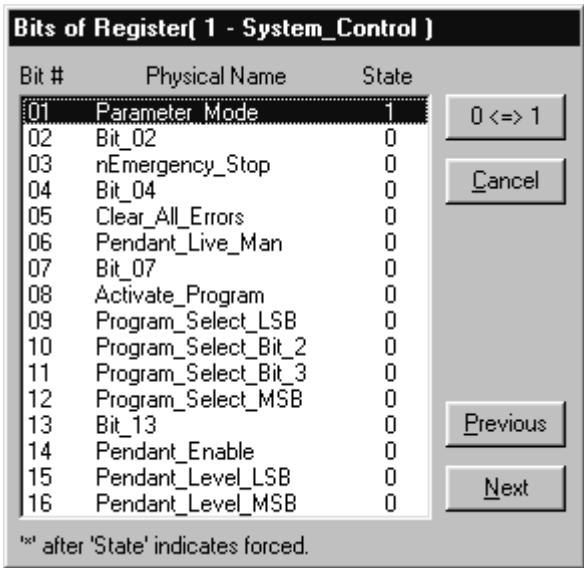
Displaying Register bits in Binary Format allows the user to easily view bit states.

Parameter Mode

Before modifications can be downloaded or updated to the CLC card, the system must be switch to Parameter Mode.

1. Select ***Data ⇒ Registers*** from VMT's main menu.
2. Double-click on *System_Control* register 001
3. Click and highlight **Bit #01**, *Parameter_Mode* and select the 

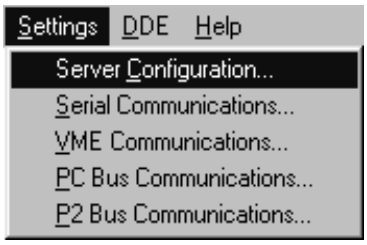
button to change the **State** of **Bit # 01** from 0 to 1.



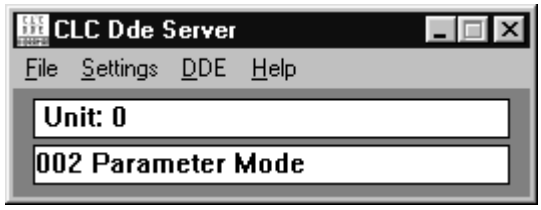
CLC DDE Server

4. The CLC card and drive should now be in Parameter Mode. To confirm the status of the system, use the CLC DDE Server. (Use Alt-Tab to display if already running)

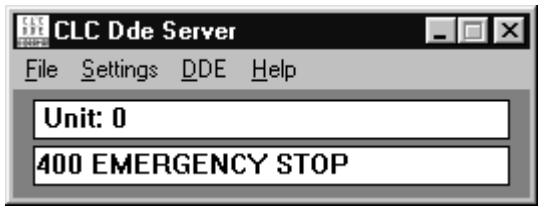
a. To display CLC system status on the DDE Server, select ***Settings ⇒ Server Configuration…*** and set CLC Status Display to **SERIAL_0** and Save.



b. If Parameter Mode was successful, the DDE Server will display…



c. otherwise, the display will read…



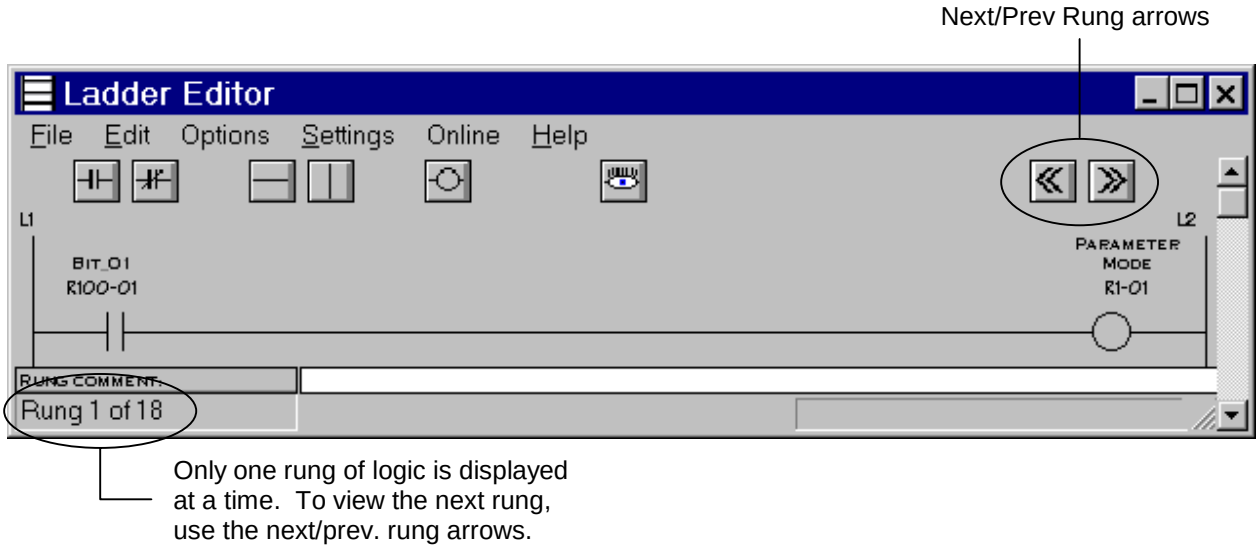
Note: If the display status of the CLC DDE Server does not change to Parameter Mode, then register 001 bit 01 is already mapped to another register. If this the case, use the following information on the I/O Mapper to correct the situation.

I/O Mapper

The I/O Mapper is displayed by selecting ***Data ⇒ I/O Mapper***. VisualMotion's I/O Mapper allows manipulation of I/O registers using Boolean strings or an optional ladder logic interface.

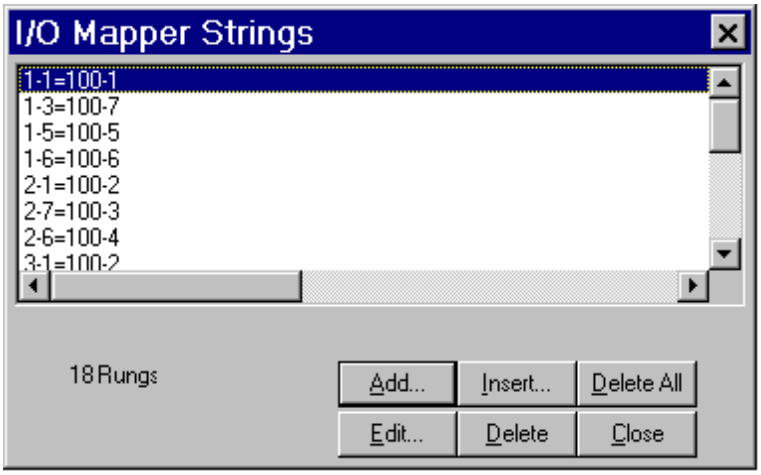
Uploading I/O Mapper

Select ***File ⇒ Upload Strings*** to view existing I/O Mapper strings currently on the CLC card.



Note: If the first rung on the I/O Mapper is blank, then the CLC card does not contain an I/O Mapper.

To view the I/O Mapper in Boolean equation form, select ***Display Strings*** from the Options menu.

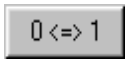


Considering the above I/O Mapper Strings, register 001 - bit 01 is map to register 100 - bit 01. In this case, the state of register 100 - bit 01 will control the Parameter Mode function of the system.

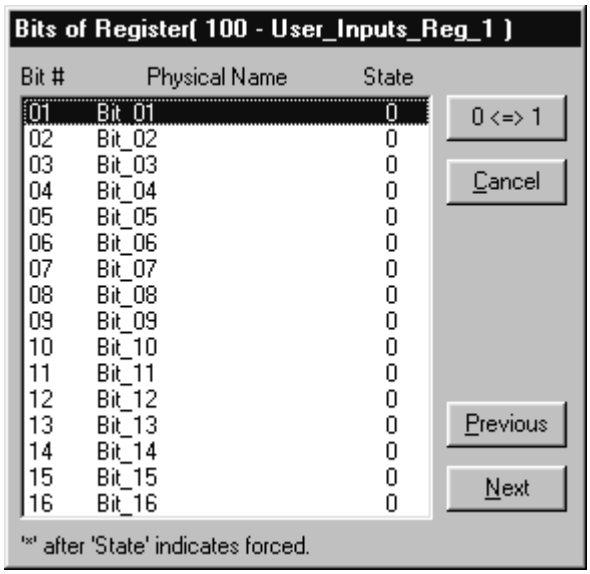
To switch the system to Parameter Mode…

1. Close all I/O Mapper windows and return to VMT's main window.
2. Select ***Data ⇒ Registers*** to open the *Active Program Registers* window.
3. Click and hold the scroll bar button and scroll down to register 100.

Note: Register numbers appear at the top of the window as you scroll down or up.

4. Click and highlight **Bit # 01** and select the 

button to change the **State** of **Bit # 01** from 0 to 1.

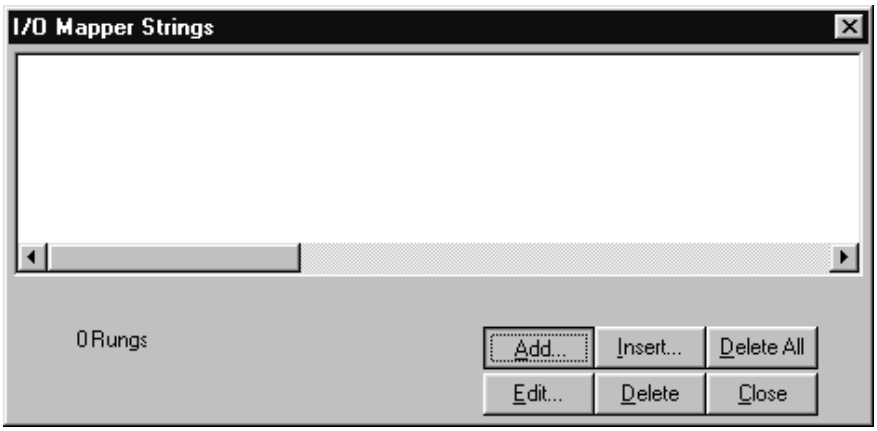


The CLC DDE Server should now display Parameter Mode.

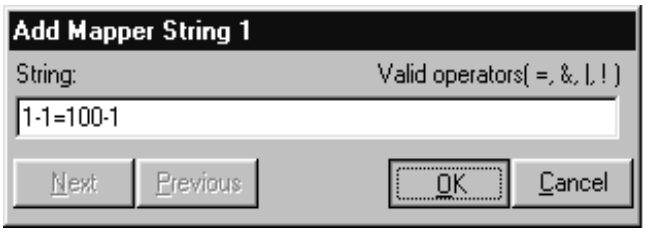
Example I/O Mapper Creation

In this example, Register 001 - bit 01, Parameter Mode, will be mapped equal to the bit state of Register 100 - bit 01.

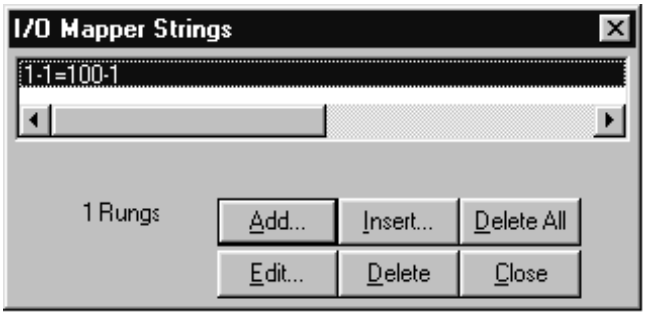
1. Select ***Data ⇒ I/O Mapper*** to open the Ladder Editor window.
2. From the Ladder Editor window, select ***Options ⇒ Display Strings***



3. Click on the **Add…** button and enter the following strings:

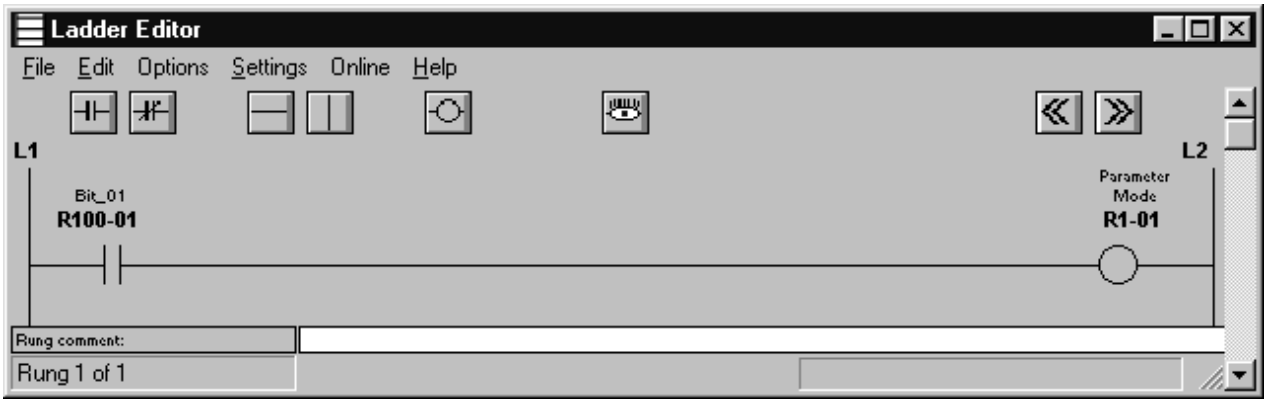


4. Click on *OK* and then *Cancel* to close the *Add Mapper String* window.



5. Now click on *Close* to return to the *Ladder Editor* window.

The Ladder Editor window now contains a Rung that illustrates Register **100-01** as an open contact and Register 1-01 as a relay coil. The symbolic relay coil can not be energized (Parameter Mode Active) until the contact **R100-1** is closed. This illustrates the relationship between Register 100 and Register 1.



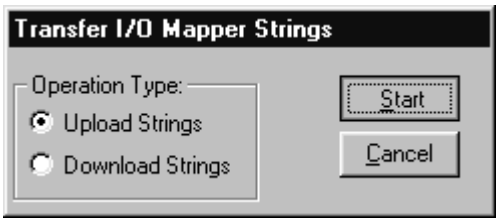
Download I/O Mapper Strings

VisualMotion installs a default I/O Mapper under the following folder directory: **C:\Indramat\clc\param**

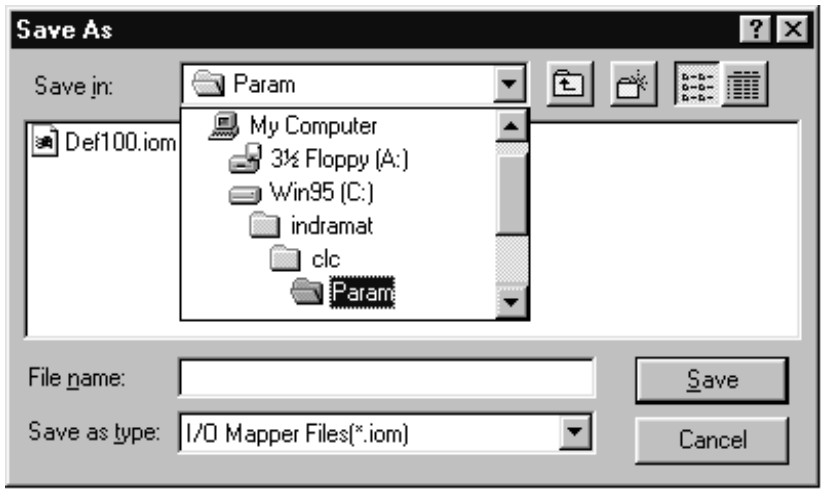
The following I/O Map can either be downloaded using the following procedure or created using the previous procedure.

Note: The system must be in Parameter Mode before proceeding.

1. From VMT's main menu select ***File ⇒ Transfer I/O Mapper***. Next, click on the Download Strings radio button and the press *Start* to begin downloading the I/O Mapper Strings.



2. Search for the harddrive path below and select the default I/O Mapper file (Der100.iom.)



The default I/O Mapper file, *Def100.iom*, is simply a text file that contains the following strings:

Actual strings found in Def100.iom

1-1=100-1	*Parameter Mode*	**4-1=100-2**	*Auto/Manual Mode Task C*
1-3=100-7	*Emergency Stop*	**4-7=100-3**	*Stop Task C*
1-5=100-5	*Clear all Errors*	**4-6=100-4**	*Start Task C*
1-6=100-6	*Pendant Live Man Switch*	**5-1=100-2**	*Auto/Manual Mode Task D*
2-1=100-2	*Auto/Manual Mode Task A*	**5-7=100-3**	*Stop Task D*
2-7=100-3	*Stop Task A*	**5-6=100-4**	*Start Task D*
2-6=100-4	*Start Task A*	**120-1=21-1**	*Parameter Mode*
3-1=100-2	*Auto/Manual Mode Task B*	**120-2=21-5**	*Error Detected*
3-7=100-3	*Stop Task B*		
3-6=100-4	*Start Task B*		

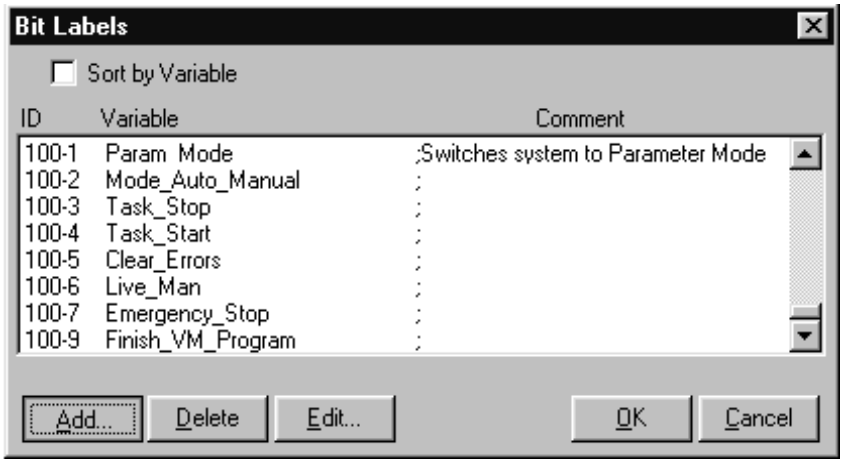
System Status Outputs can be used for external monitoring

String description not found in file

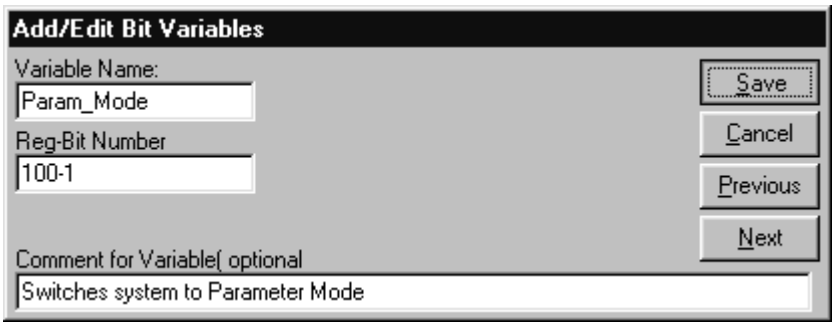
Bit Labels

Bit labels are names given to I/O Mapper strings making them easier to identify and use. Bit label information is saved with each VisualMotion Toolkit program file. Once the following labels have been added, the sample program must be saved, compiled, downloaded and activated on the CLC card.

1. Select ***Labels ⇒ Bit Labels...*** under the Edit menu.



2. Select *Add...* and enter the labels shown above for **Reg-Bit Number 100-1** to **100-9**.

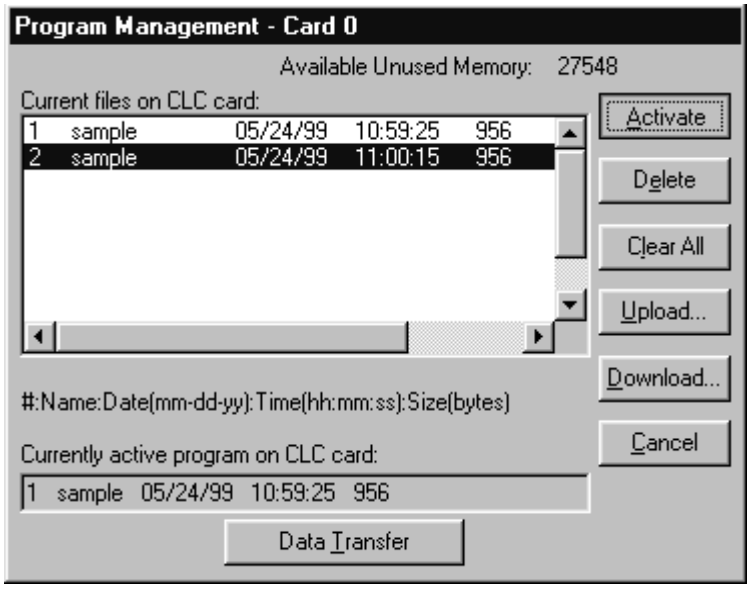


3. After adding the last bit label, click on *Cancel* to close the *Add/Edit Bit Variables* window. Now, click on the *OK* button to close the Bit Labels window.

4. Select **Save, Compile, Download** from the file menu or select the icon from the Toolbar.

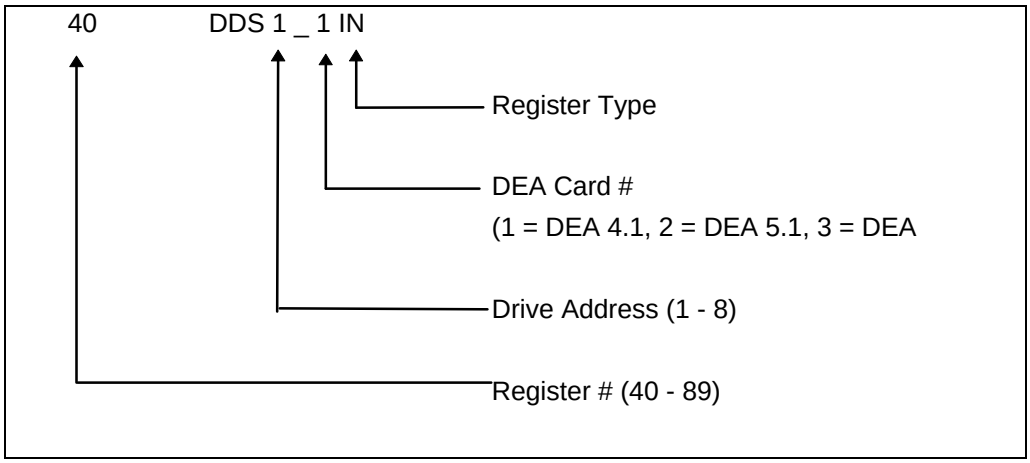
5. After the download is complete, select File Program Management and click on the Activate button. To activate a different program, highlight the file and Activate.

Note: If a program is Activate while a different program is currently running, an error will be issued by VisualMotion Toolkit. Stop all motion before activating a new program.



Creating an I/O Map for a DEA Card

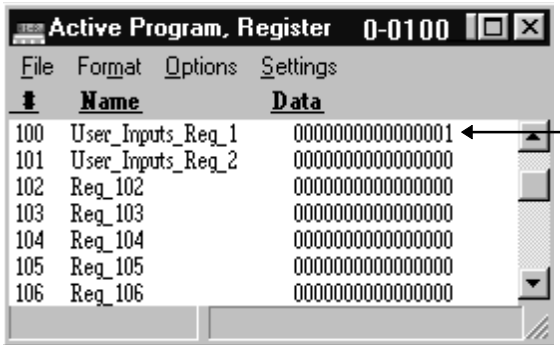
1. Determine the correct register for your DEA card.
   a) Refer to Appendix C. Non-CLC System Hardware - *CLC VME I/O Systems*.
   b) Refer to the **VisualMotion 6.0 Reference Manual**:
   Chapter 2. CLC I/O Systems - Registers 40-89.
   Chapter 4. VisualMotion Menu Commands - **I/O Setup** under the **Setup** Menu.
   c) Even numbered registers are input registers and odd numbered registers are output registers.



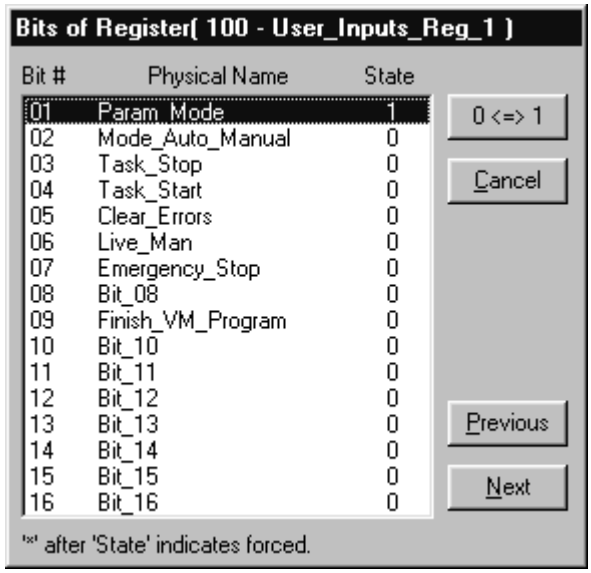
4.3 Run Sample Program

The following procedure covers program execution and operation. Before following this procedure be sure that the initial setup covered in the previous section (***Initial Setup Prior to Operation***) has been completed. The sample program should be open within VisualMotion Toolkit and activated on the CLC card.

1. Select Data Registers from the main menu. Scroll down to register 100 and double-click on it to open.

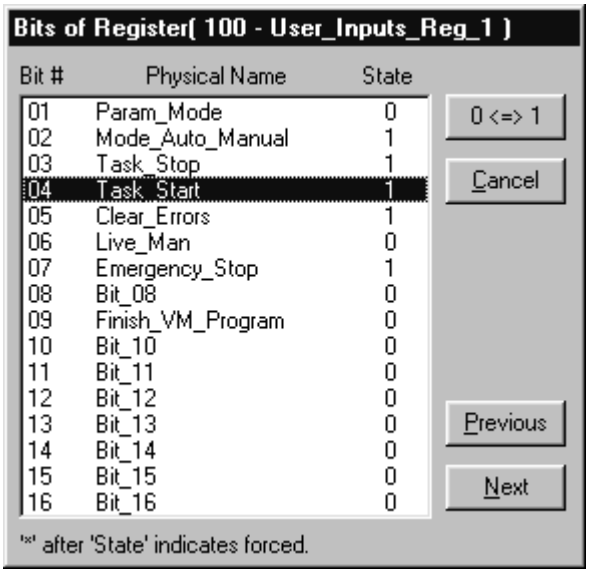


The bits should now be listed along with their corresponding labels (Physical Name.)



2. Change the state of the following bits as listed in the order shown below:

	Bit	Label	State
a.	07	Emergency-Stop	0 to 1
b.	01	Param_Mode	1 to 0
c.	05	Clear_Errors	0 to 1
d.	02	Mode_Auto_Manual	0 to 1
e.	03	Task_Stop	0 to 1
f.	04	Task_Start	0 to 1

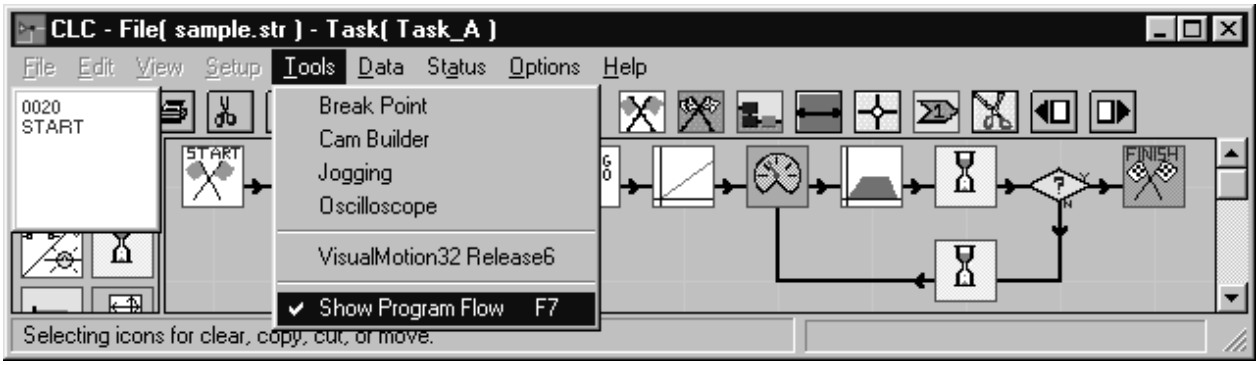


3. The program should now be running with the bits in this state.

Note: If variables were added to the sample program make sure that they have been assigned a value before starting the program. ***See Chapter 3. Assigning a Value to a Variable***.

Note: If the program has been started prior to adding values to the variables, the program will not run because all variable values are zero. Stop the program by changing the state of bits 3 and 4 from 1 to 0. Once values have been added, reinitialize the program by changing the state of bits 3 and 4 back to 1.

To view which program icon is being processed within VisualMotion Toolkit, select ***Tools ⇒ Show Program Flow*** or press **F7** on the keyboard to turn the feature on or off.



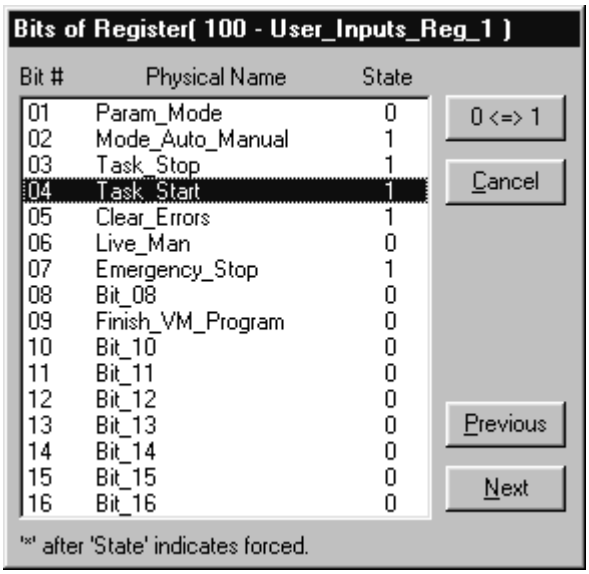
Program Operation

Changing the state (0 <=> 1) of any one of the following bits in Register 100 will stop the program from running.

Bit	Label	State
01	Param_Mode	1
02	Mode_Auto_Manual	0
03	Task_Stop	0
07	Emergency-Stop	0
09	Finish_VM_Program	1

If the program was stopped using bit 100-7, *Emergency Stop,* or from another error, bit 100-5 *Clear_Errors* must be toggled from 1 to 0 to 1 before running again.

To run the program again, toggle bit 100-4 *Task Start* from 1 to 0 to 1 with the other bits in the state shown below:



Note: In order to start the program from the very begining, toggle bit 100-2, Mode_Auto_Manual, from 0 to 1 prior to toggling the Task Start bit.

4.4 Parameter Overview

There are four sets of parameters contained within a CLC/Drive system: Card (C), Task (T), Axis (A) and Drive (S, P) parameters. Each parameter can be viewed and edited, where applicable, from VisualMotion. Axis, Card and Task parameters are stored on the CLC card while the Drive parameters are stored in the DSM firmware module on the drive.

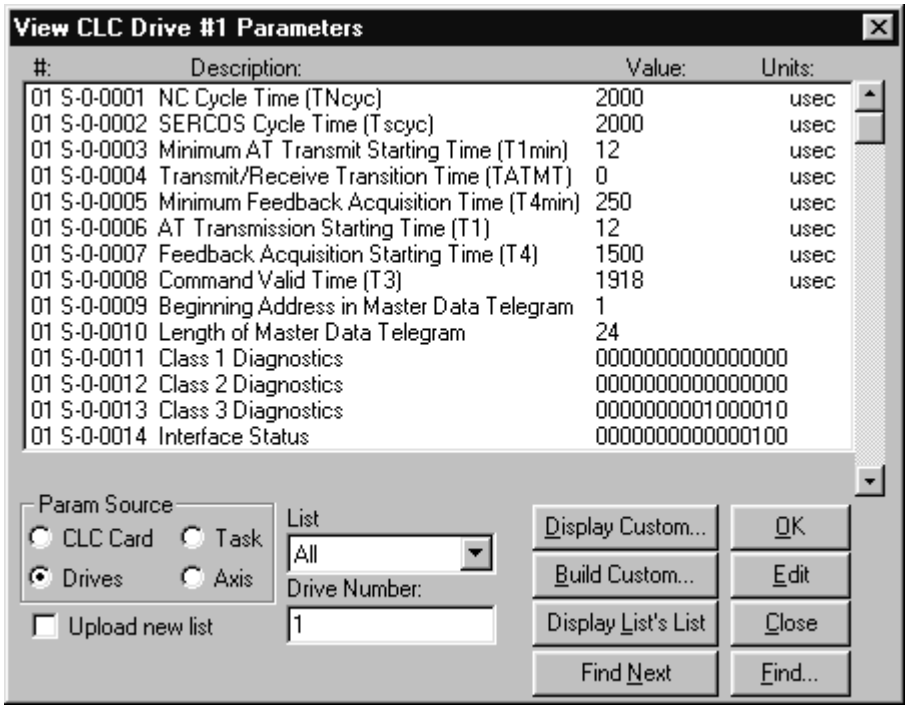
Many of the basic parameter values needed for initialization will be present with a new system. However, because of the wide variety of configurations available for each application, some of the these parameters may need to be modified by the system-builder before the system will initialize properly. See the table below:

S-0-0057	In Position Window	0.1in, 1deg, or 1mm
S-0-0091	Bipolar Velocity Limit Value This is a software limit on the maximum velocity of the **drive**. It must be ≤ maximum motor velocity.	(see S-0-0113)
S-0-0092	Bipolar Torque Limit Value	400%
S-0-0103	Modulo Value (= 1 motor rev. for startup program) Rotary mode or linear positioning with modulo only.	1 in, 360 deg, or 10 mm
S-0-0124	Standstill Window (Zero Velocity - DIAX02)	0 < rpm < 10
S-0-0159	Monitoring Window (100% = 360°)	25%
P-0-0004	Smoothing Time Constant	250 µs
S-0-0349	Bipolar Jerk (P-0-0106 - DIAX02)	2000000
A-0-0020	Maximum Velocity This is a software limit on the maximum **axis** velocity for this Task. This must be ≤ Bipolar Velocity Limit of the drive. Axis velocities greater than this value will cause errors.	(see S-0-0091)
A-0-0021	Maximum Acceleration This is a software limit on the maximum **axis** acceleration for this Task.	(see S-0-0138)
A-0-0022	Maximum Deceleration This is a software limit on the maximum **axis** deceleration for this Task.	(see S-0-0138)

The CLC Reference Manual provides a detailed listing of the Axis, Card, and Task parameters. A listing of the Drive parameters can be found in the respective drive user's manual.

Viewing Parameters

In VisualMotion Toolkit, select “Overview” from the “Setup” menu.


View CLC Drive #1 Parameters
#: Description: Value: Units:
01 S-0-0001 NC Cycle Time (TNcyc) 2000 usec
01 S-0-0002 SERCOS Cycle Time (Tscyc) 2000 usec
01 S-0-0003 Minimum AT Transmit Starting Time (T1min) 12 usec
01 S-0-0004 Transmit/Receive Transition Time (TATMT) 0 usec
01 S-0-0005 Minimum Feedback Acquisition Time (T4min) 250 usec
01 S-0-0006 AT Transmission Starting Time (T1) 12 usec
01 S-0-0007 Feedback Acquisition Starting Time (T4) 1500 usec
01 S-0-0008 Command Valid Time (T3) 1918 usec
01 S-0-0009 Beginning Address in Master Data Telegram 1
01 S-0-0010 Length of Master Data Telegram 24
01 S-0-0011 Class 1 Diagnostics 0000000000000000
01 S-0-0012 Class 2 Diagnostics 0000000000000000
01 S-0-0013 Class 3 Diagnostics 0000000001000010
01 S-0-0014 Interface Status 0000000000000100
Param Source
CLC Card
Task
Drives
Axis
Upload new list
List
All
Drive Number:
1
Display Custom...
Build Custom...
Display List's List
Find Next
OK
Edit
Close
Find...

In the “Param Source” box, choose the type of parameter to view, either Axis, Card, Drive or Task. For Axis and Drive parameters, it will be necessary to specify the axis/drive number. For Task parameters, it will be necessary to specify the Task letter (A, B, C, or D). After choosing the source of the parameter list, click the “OK” button.

Editing Parameters

Step 1.	Upload the type of parameter to edit, either Axis, Card, Drive or Task.
Step 2.	Highlight the desired parameter and double click or click the “Edit” button. *Hint*: When using the scroll bar in the parameter list, the parameter number at the top of the screen will appear in the upper left right corner of the Overview box..
Step 3.	Enter the new parameter data in the “Edit Parameter” dialog box. “Save” the change and choose “Cancel” to close the box. *Hint*: Some parameters can only be changed while the system is in parameter mode. VisualMotion Toolkit will warn you when trying to change those parameters.

Note: Before editing any VisualMotion system parameter, stop any running programs and switch the system to Parameter Mode. **Parameter values should always be performed by trained personnel.**

Building Custom Parameter Lists

Within each type of parameter, it is possible to build a custom list that will display only the parameters the system-builder chooses.

Step 1.	Upload the entire list of parameters from the desired source.
Step 2.	Click the “Build Custom...” button. The “Build Custom Display” dialog box will open.
Step 3.	Highlight a parameter from the parent list to include in the custom list and click the “Add” button. Repeat this step until list is complete.
Step 4.	Click “Save” and enter an 8 character name for the list.
Step 5.	“Cancel” to close the “Build Custom Display” dialog box.
Step 6.	Click “Display Custom...” and choose the custom list from the “Display Custom List” dialog box.

5 DDE Client Interfaces

The following examples illustrate how to create custom DDE client interfaces for the CLC. DDE communication allows certain software applications to read from (*request*), and write to (*poke*) the CLC card. **Refer to the *VisualMotion 6.0 Trouble Shooting Guide, Chapter 4. CLC DDE SERVER* for more information.**

5.1 Creating and Customizing a DDE Client Interface in Microsoft Excel™

The following example illustrates how to create a custom DDE client interface for the CLC card using Microsoft Excel™ (*Version 5.0 and up*) Other programs that support DDE communication can also be used in a similar fashion. Requested information can be read directly by a spreadsheet, chart or database, while poke transactions allow users to control program execution from within this custom interface.

The example below was created for an ELS/CAM application. The instructions for creating the CLC program that corresponds with this spreadsheet can be found in the *Appendix E. Example Programs*. All of the *requested* data in the spreadsheet was read from the CLC card with the CAM program running.

DDE Worksheet Functions

A DDE request can be made directly from a *cell* within an Excel™ Worksheet using a formula outlined below. The CLC_DDE Server should be running before a request is made. Each request is queued in the server and then handled using round robin arbitration. The Excel Worksheet will automatically update the cell as the information becomes available. The response time varies according to how many other applications are running and how many DDE conversations are occurring at the same time. Limit the number of active DDE requests within a worksheet in order to get a faster response time.

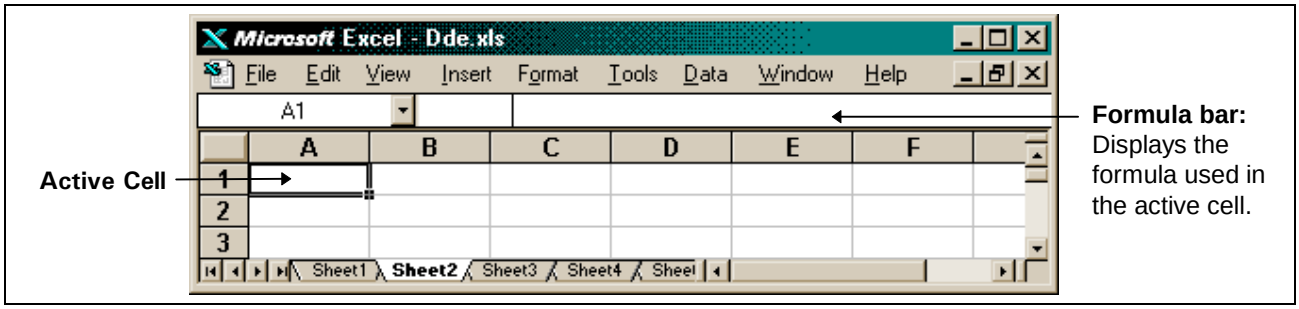
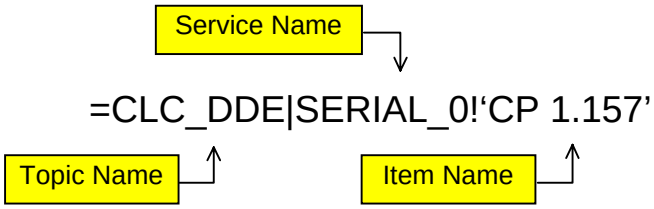


Figure 5-1: Excel Worksheet

Select a cell and then enter the **DDE Service name**, **Topic name** and **Item name** within the formula bar using the following syntax as an example:



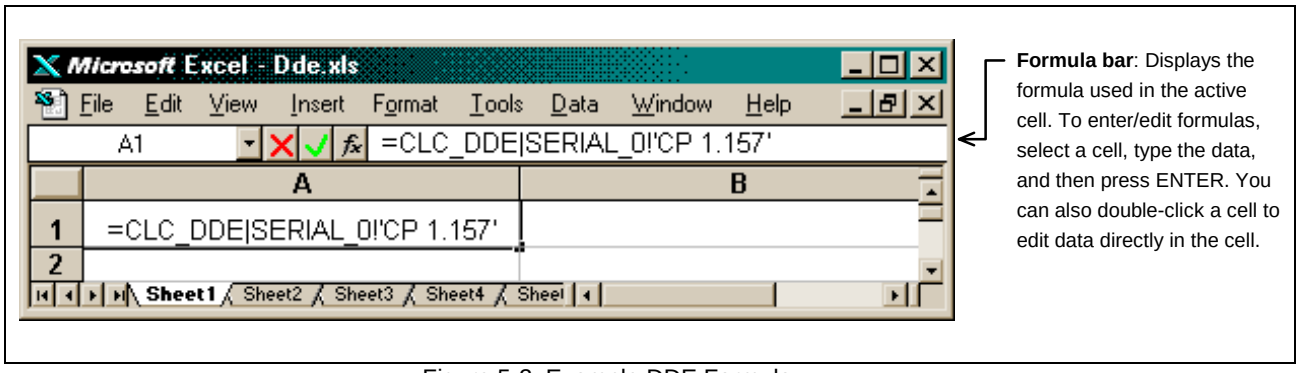

Microsoft Excel - Dde.xls
File Edit View Insert Format Tools Data Window Help
A1
=CLC_DDE|SERIAL_0!'CP 1.157'
A
B
1
=CLC_DDE|SERIAL_0!'CP 1.157'
2
Sheet1 Sheet2 Sheet3 Sheet4 Sheel
Formula bar: Displays the formula used in the active cell. To enter/edit formulas, select a cell, type the data, and then press ENTER. You can also double-click a cell to edit data directly in the cell.

Figure 5-2: Example DDE Formula

The **Item name** 'CP 1.157' will read the value of the system or "Card" Parameter C-0-0157 which is the current ELS master position.

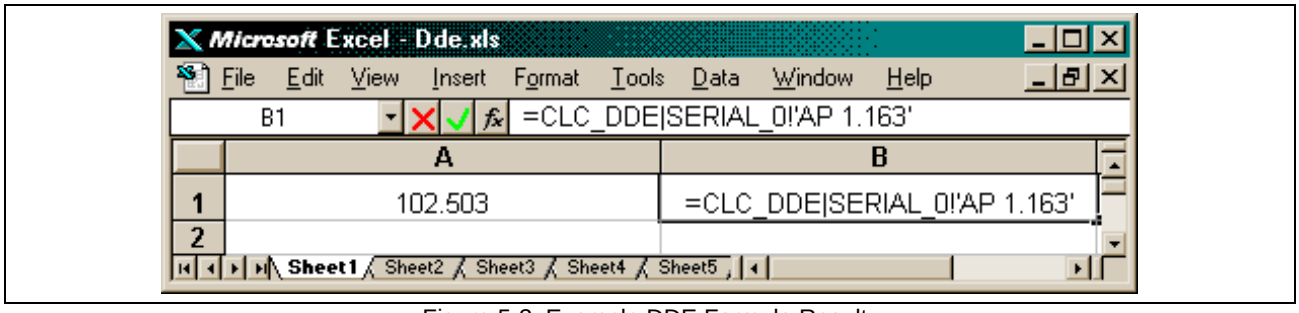
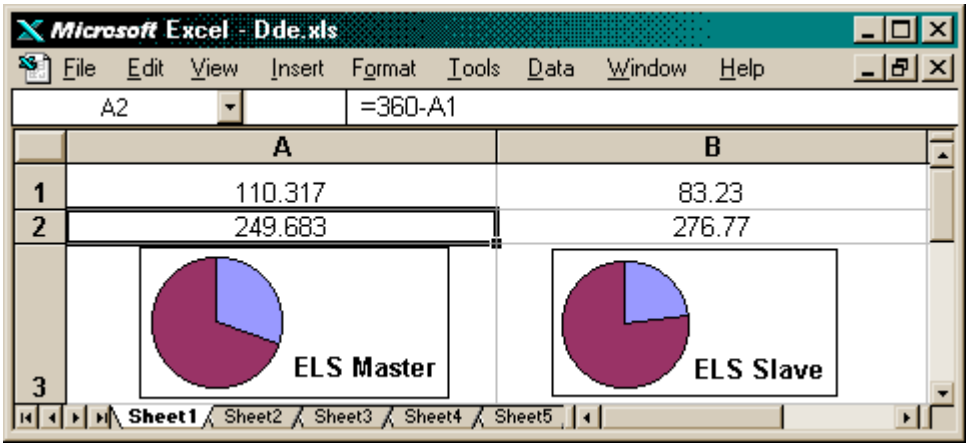


Figure 5-3: Example DDE Formula Result

The **Item name** 'AP 1.163' will read the value of the Axis Parameter A-0-0163 which is the CLC Cam Output Position. **Refer to the *VisualMotion 6.0 Reference Manual, Appendix A. Direct ASCII Communications,* for the syntax used with other item names.**

A pie chart can be used to graphically illustrate the master and slave positions between 0° and 360°. Enter two formulas in the second row which subtract the ELS Master and Slave Positions from 360° (=360-A1,=360-B1). Select each column and create a pie chart using the Excel *Chart Wizard*. If the CLC_DDE Server is active and an ELS program is running the pie charts will rotate to reflect the current ELS positions as they change.

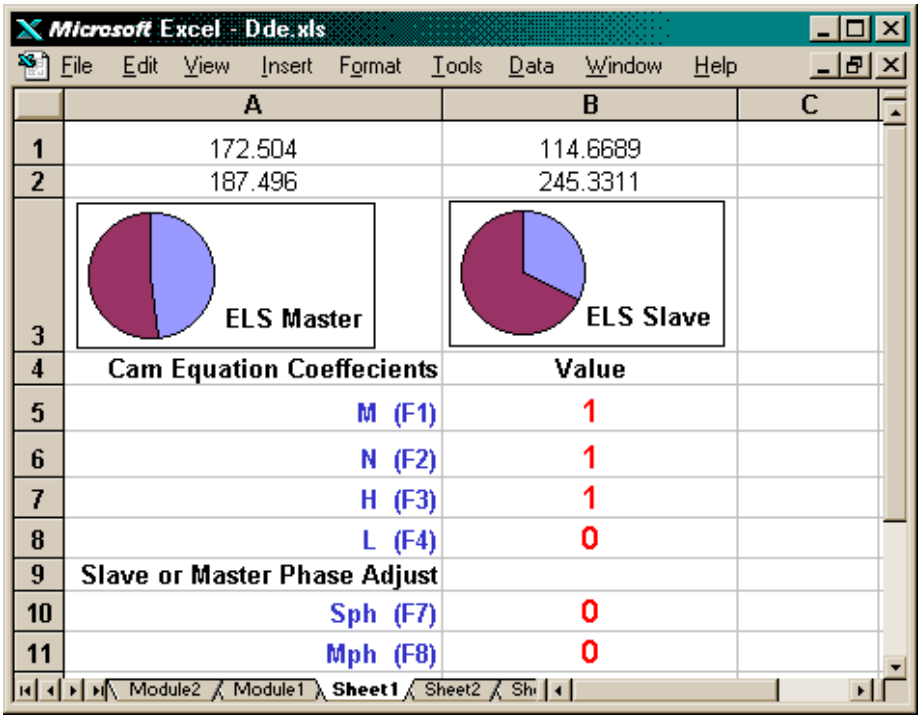


DDE Functions using Visual Basic® for Excel™

Visual Basic® for Excel™ has its own DDE Functions which can be used in a spreadsheet macro or module. The following Visual Basic® macros illustrate how to use the *DDERequest* and *DDEPoke* functions. The *DDERequest* function is used to read the values of the CAM coefficient and phase offset parameters. The *DDEPoke* function is used to write values to predefined program variables which were added in the spreadsheet. Each variable also has a corresponding **Item name** needed for DDE communication.

The variables listed in column A were predefined in the CAM program to store the CAM coefficients and Phase Adjust values. **Macro 1** requests the current coefficient values from the corresponding CLC System (card) and Axis parameters.

Note: *When making a DDERequest from a macro the Service name and Topic name are included in the DDEInitiate function and assigned to a variable (MYCLC*).


Microsoft Excel - Dde.xls
File Edit View Insert Format Tools Data Window Help
ELS Master
ELS Slave
Module2 Module1 Sheet1 Sheet2

	A	B	C
1	172.504	114.6689	
2	187.496	245.3311	
3			
4	Cam Equation Coeffecients	Value	
5	M (F1)	1	
6	N (F2)	1	
7	H (F3)	1	
8	L (F4)	0	
9	Slave or Master Phase Adjust		
10	Sph (F7)	0	
11	Mph (F8)	0	

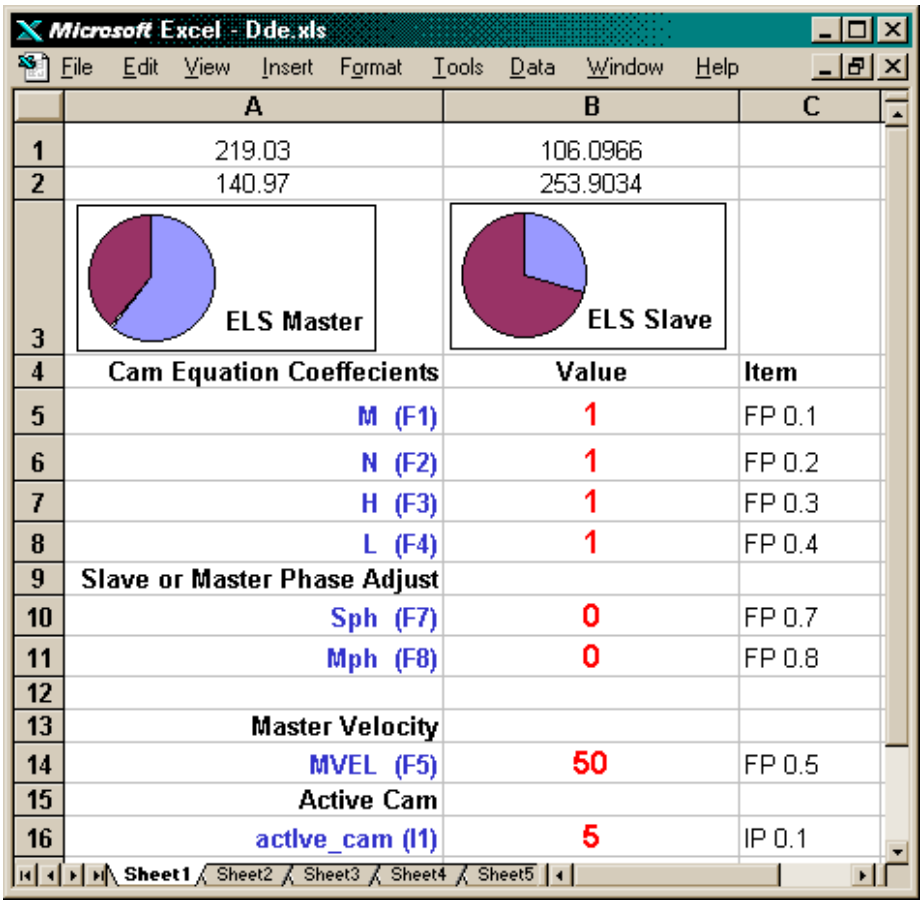
```
Function Request()                                          DDEInitiate ("Service Name", "Topic Name")
MYCLC = DDEInitiate("CLC_DDE", "SERIAL_0")

m = Application.DDERequest(MYCLC, "AP 1.032")               A-0-0032 Cam Slave Factor (M)
Worksheets(1).Cells(5, 2).Value = m
n = Application.DDERequest(MYCLC, "AP 1.031")               A-0-0031 Cam Master Factor (N)
Worksheets(1).Cells(6, 2).Value = n
h = Application.DDERequest(MYCLC, "AP 1.033")               A-0-0033 Cam Stretch Factor (H)
Worksheets(1).Cells(7, 2).Value = h
l = Application.DDERequest(MYCLC, "AP 1.035")               A-0-0035 Cam Master Position (L)
Worksheets(1).Cells(8, 2).Value = l
sph = Application.DDERequest(MYCLC, "AP 1.162")             A-0-0162 Cam Slave Phase Adjust
Worksheets(1).Cells(10, 2).Value = sph
mph = Application.DDERequest(MYCLC, "AP 1.151")             A-0-0151 ELS Phase Offset
Worksheets(1).Cells(11, 2).Value = mph

End Function
```

Figure 5-4: Macro 1

Additional variables were added to column A to adjust the ELS Master velocity (F5) and the active CAM number (I1). Column C contains the **Item name** which corresponds with each program variable (F1-F8 and I1). The DDEPoke command in **Macro 2** references the worksheet for the DDE **Item name** and value for each variable.


Microsoft Excel - Dde.xls
File Edit View Insert Format Tools Data Window Help
ELS Master
ELS Slave
Sheet1 Sheet2 Sheet3 Sheet4 Sheet5

	A	B	C
1	219.03	106.0966	
2	140.97	253.9034	
3			
4	Cam Equation Coeffecients	Value	Item
5	M (F1)	1	FP 0.1
6	N (F2)	1	FP 0.2
7	H (F3)	1	FP 0.3
8	L (F4)	1	FP 0.4
9	Slave or Master Phase Adjust		
10	Sph (F7)	0	FP 0.7
11	Mph (F8)	0	FP 0.8
12			
13	Master Velocity		
14	MVEL (F5)	50	FP 0.5
15	Active Cam		
16	actlve_cam (I1)	5	IP 0.1

When **Macro 2** is executed the data in the value column will be written or "poked" to the variables defined by the corresponding **Item names**. This allows the user to see how different values will alter the performance of

the slave axis with respect to the master. The DDEPoke command uses the following syntax:

Application.DDEPoke *MYCLC*, *Item Name*, *Value*

MYCLC is defined in the *DDEInitiate* command and includes the DDE **Service name** and **Topic name.**

```
Function Poke()

MYCLC = DDEInitiate("CLC_DDE", "SERIAL_0")

Application.DDEPoke MYCLC, Worksheets(1).Cells(5, 3).Value, Worksheets(1).Cells(5, 2)
Application.DDEPoke MYCLC, Worksheets(1).Cells(6, 3).Value, Worksheets(1).Cells(6, 2)
Application.DDEPoke MYCLC, Worksheets(1).Cells(7, 3).Value, Worksheets(1).Cells(7, 2)
Application.DDEPoke MYCLC, Worksheets(1).Cells(8, 3).Value, Worksheets(1).Cells(8, 2)
Application.DDEPoke MYCLC, Worksheets(1).Cells(10, 3).Value, Worksheets(1).Cells(10, 2)
Application.DDEPoke MYCLC, Worksheets(1).Cells(11, 3).Value, Worksheets(1).Cells(11, 2)

Application.DDEPoke MYCLC, Worksheets(1).Cells(14, 3).Value, Worksheets(1).Cells(14, 2)

Application.DDEPoke MYCLC, Worksheets(1).Cells(16, 3).Value, Worksheets(1).Cells(16, 2)

End Function
```

Figure 5-5: Macro 2

5.2 Wonderware

In order for Wonderware to communicate with the CLC, a DDE link between the two must be created. The link, or DDE Access, tells Wonderware what Windows application to use (clc_dde.exe server) in order to communicate with the CLC. This application must be running in order for Wonderware to communicate with the CLC.

1. To establish a DDE link:
   a. Choose DDE Access Names under the Special Menu in Intouch Development. The DDE Access Name Definition window will open
   b. Press the Add button:

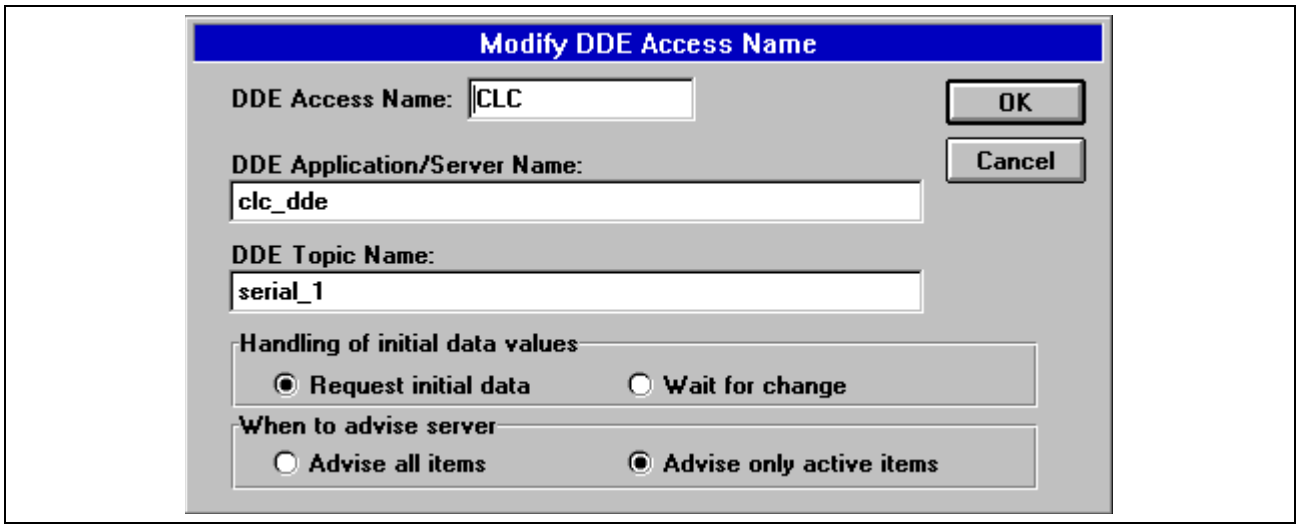


Figure 5-6: DDE Access Name

- DDE Access Name:

  The DDE Access Name can be any name you choose.

- DDE Application/Server Name:

  For the CLC, this is the clc_dde server (clc_dde.exe) provided in the VisualMotion Toolkit. It is not necessary to use the *.exe* extension. It is good practice, however, to include the path for the clc_dde server in the DOS Path statement and to configure Wonderware to launch the clc_dde server. If Wonderware is in your Windows Start-up group, then the clc_dde server application should also be in that group. Since Windows launches applications in the startup group from left to right, the clc_dde server icon should be to the left of the Intouch icon.

- Topic Name:

  The topic name will depend upon the method of communication between the computer and the CLC. The following items describe the different methods of communication.

  Serial Communications

  If you are communicating with the CLC (D, P, or V) via the computer's serial port, the topic name will be "serial_*x*" where *x* is the CLC device number, card parameter C-0002. The default for a CLC-D and CLC-P is device #0. The CLC-V is switch selectable via the Mode switch on the front of the card.

  PC Backplane Communications

  If you are communicating with the CLC-P via the PC backplane, the topic name will be "isa_*x*" where *x* is the PC address of the card.

  VME Backplane Communications

  If you are communicating with the CLC-V via the VME backplane, the topic name will be "xycom_*x*" where *x* is the VME card address set by the Mode switch on the front of the CLC-V.

  If you are still not sure of the topic name, it can be found in the CLC server application after VisualMotion has established communications with the CLC. To find the active topic name, open "Server Configuration" under the Settings menu in the clc_dde server application. The topic name is found in the "CLC Status Display" box.

Tagnames

To display a CLC parameter, variable, etc. in Wondware, the following type of tagname can be used:

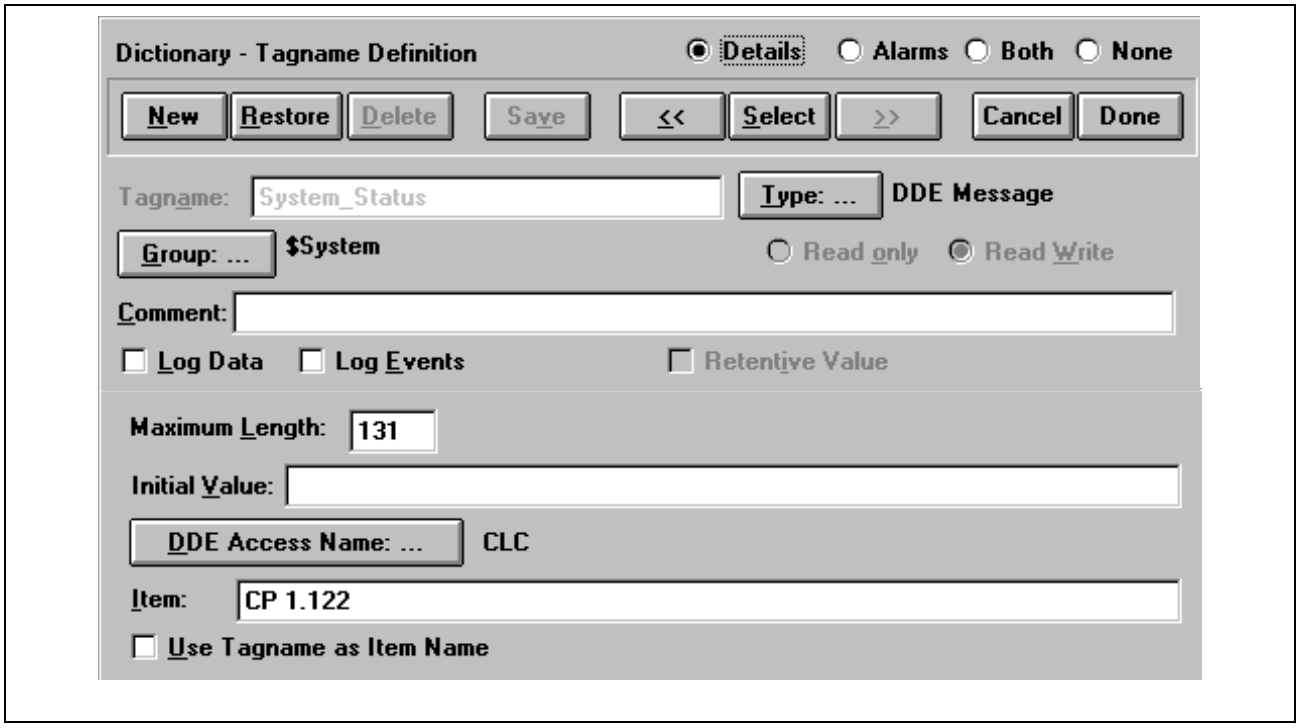


Figure 5-7: Tagname for Displaying Parameters

The tagname in Figure 5-7: Tagname for Displaying Parameters is labeled *System_Status*. It is type DDE Message, since it is only displaying the parameter. DDE Access name is “CLC”, from Figure 5-6: DDE Access Name.

The item field requests the parameter or variable to display. In Figure 5-7: Tagname for Displaying Parameters, the item requests CLC Card Parameter 122, system staus. Other examples:

Parameters	**Example**	
Card	CP 1.122	Card param. 122 System Status Message
Task	TP 1.123	Task A param. 123, Task Status Message
Drive	DP 2.95	Drive 2, param 95 Drive Status Message
Axis	AP 3.4	Axis 3, param. 4 Axis Options
Variables	**Example**	
Floating Points	FP 0.12	Active Program, float # 12
Integers	IP 1.5	Program #1, integer #5
Global Floats	GF 0.1	Active Program, global float #1
Global Integers	GI 2.2	Program #2, global integer #2

To write to a CLC variable, parameter, etc, the following type of tagname can be used:

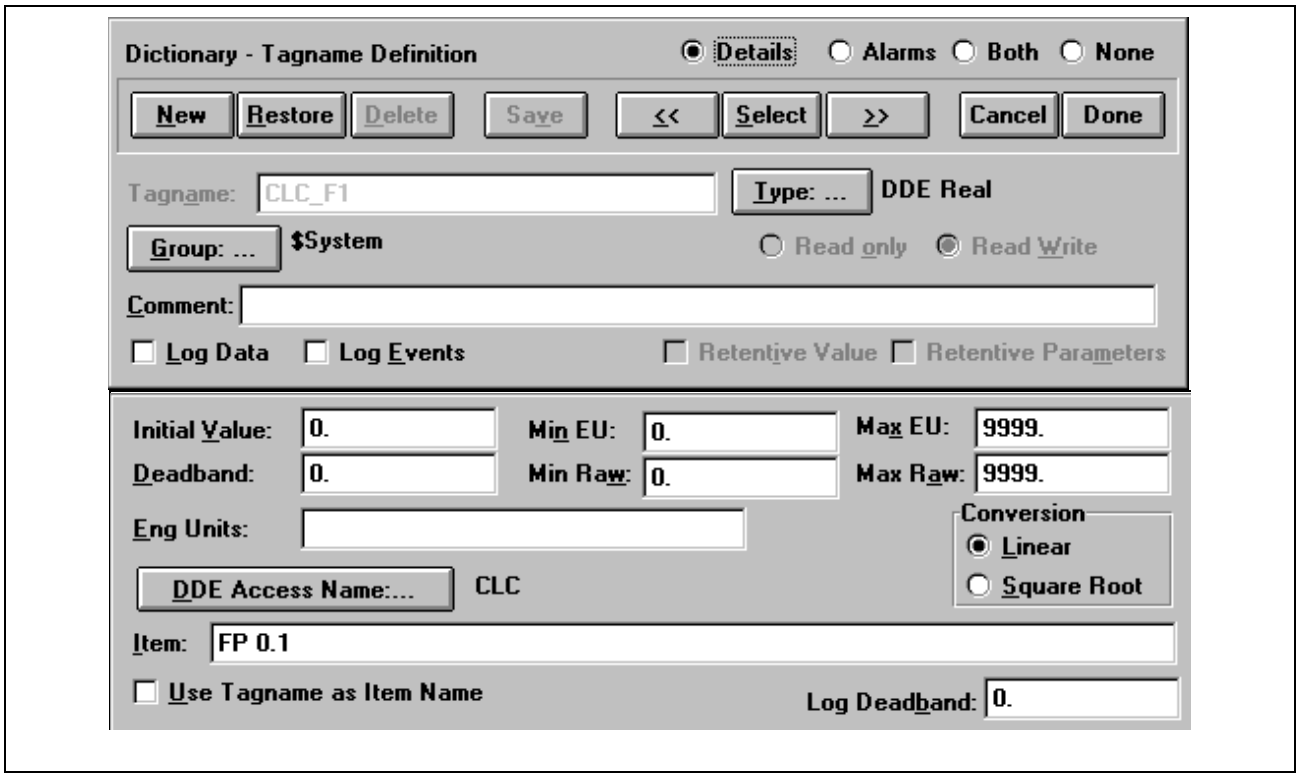


Figure 5-8: Tagname for Writing to Variables

The tagname in Figure 5-8: Tagname for Writing to Variables makes floating point variable #1 of the active CLC program a DDE real variable. The value of FP 0.1 can be changed in Wonderware by changing the value of the tagname, CLC_F1.

To change a bit in a CLC register:

Configure a tagname as a Type DDE Integer, min EU -99999, min Raw -99999

max EU 99999, max Raw 99999

Tagname Reg_100

Item RD 0.100

The above tagname will write to register 100. To change a specific bit in that register, configure a pushbutton as shown below.

Object type: Button

Button Type: User Input, Discrete

Tagname: Reg_100.00

The above button has two states, on and off. When in the on state, register 100 bit 1 will be set to 1. When in the off state, that bit will be set to 0. To write to bit 2, change the tagname to Reg_100.01, bit 3 = Reg_100.02, bit 4 = Reg_100.03, ... bit 16 = Reg_100.15.

A PC Interface (CLC-P: ISA - PC/104 bus)

A.1 General information

The CLC-P02 communications to a PC via a 4Kword (4K x 16 bit) area of the dual port RAM.

Four buffers are available for data exchange. Variable and register data can be directly read while ASCII communication is controlled via interrupts and the contents of the associated DPRAM register.

The following relationship exists between the PC and the CLC-P.

- When parameters are transmitted via the ASCII Protocol, the PC requests transmission and the CLC-P acknowledges after executing the transmission. This method is used to ensure data consistency.
- Global Integers, Global Floats and I/O Registers can be read or written at anytime.

Note: Care must be taken to ensure data consistency in the CLC user program.

- For the PC to be able to recognize a CLC-P, the CLC writes an identification reset into the DPRAM after switch on.

A.2 Base Address

CLC-P01 The base address of the DPRAM on the CLC-P01 card is set by jumpers S8, S9, S10 and S11.

Sixteen (16) different base addresses can be set in 16K increments over the address range 000C0000 to 000FC000. Refer to *Table 2-1: CLC-P01 card number and base memory address selection* for details.

CLC-P02 The base address of the DPRAM on the CLC-P02 card is set by opening and closing switches 1 thru 4 on the S1 DIP switch.

Sixteen (16) different base addresses can be set in 8K increments over the address range 000D0000 to 000EE000. Refer to *Table 2-2: PC/104 card base memory address selection* for details.

A.3 Bus Communications

CLC-P01 (ISA bus)

The CLC-P01 uses a 4 Kbytes dual port RAM to exchange information between the CLC card and the host PC. On GPS firmware, the dual port RAM is split into several areas, which is accessible from both the CLC-P01 and the host computer's ISA bus.

CLC-P02 (PC/104 bus)

Although the CLC-P02 hardware can support an 8 Kbytes dual port RAM area, GPS 6 firmware utilizes 4 Kbytes to exchange information between the CLC card and the host PC. On GPS firmware, the dual port RAM is split into several areas, which is accessible from both the CLC-P02 and the host computer's PC/104 bus. Some fields in the dual port RAM can be used for communications with a PLC or another real-time controller.

Dual Port RAM Organization

The address offset in *Table A-1: Dual Port RAM Organization* is added to the card's base address.

The following is an example of the Comms Buffer memory location for card number 4 of the CLC-P01/P02.

CLC-P01

		Start	End
Comms Buffer	=	0xC00	0xCFF
Base Address	=	D0000	D0000
		D0C00	D0CFF

The Comms Buffer for card 4 of the CLC-P01starts at D0C00 and ends at D0CFF

CLC-P02

		Start	End
Comms Buffer	=	0xC00	0xCFF
Base Address	=	D8000	D8000
		D8C00	D8CFF

The Comms Buffer for card 4 of the CLC-P02 starts at D8C00 and ends at D8CFF

The CLC-P uses a 4 Kbytes dual port RAM to exchange information with the Host PC. The dual port RAM is split into several areas, as shown in the following figure, which are all accessible from both the CLC-P card and the PC's ISA bus. The CLC reads bits in Motorola format.

There are 256 global float and 256 global integer variables, and there are 512 16-bit registers. Additionally, there is a response flag for the Comms_Buffer, as well as interrupt flags. At this time, the CLC-P does not interrupt the PC. The Comms_Buffer is 256 bytes. It uses the same protocol as serial communications. The flowchart below depicts the handshaking protocol in the dual port RAM and can be used for direct access to the Comms_Buffer when not using the DDE Server.

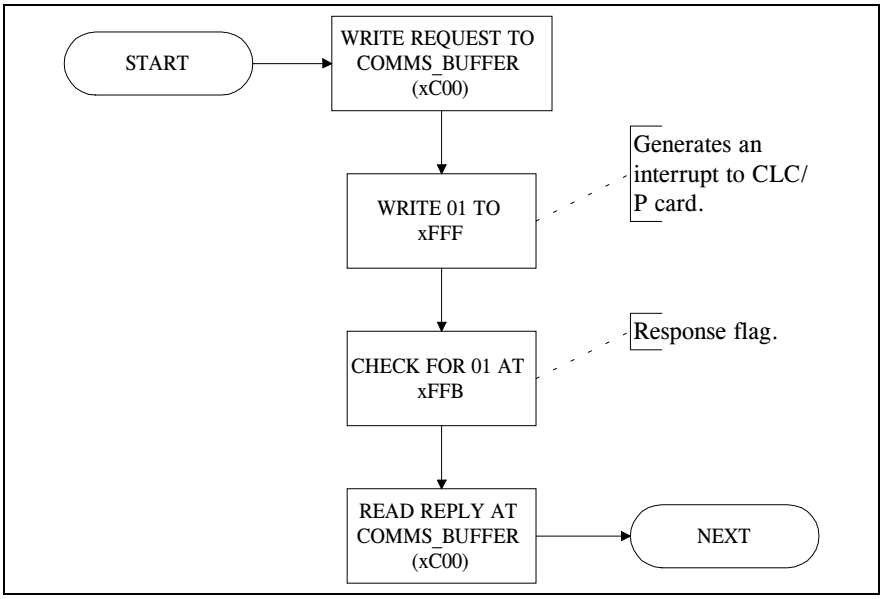


Figure A-1: Dual Port RAM Handshaking Protocol

Data	Format	CLC-P01 Memory Range		CLC-P02 Memory Range	
		Start	End	Start	End
Global Integers(GIx)	int_32	0x000	0x3FF	0x000	0x3FF
Global Floats(GFx)	real_32	0x400	0x7FF	0x400	0x7FF
Registers(Rx)	int_16, Boolean	0x802	0xBFF	0x802	0xBFF
Comms Buffer	VM ASCII	0xC00	0xCFF	0xC00	0xCFF
Status and Control Bits					
Status Bits	Boolean	0xD00	0xD00	0xD00	0xD00
ID String 1	ASCII	0xD04	0xD0A	0xD04	0xD0A
CLC ID String	ASCII	*not available*	*not available*	0xF40	0xF4F
Master ID String	ASCII	*not available*	*not available*	0xF50	0xF5F
Time	32 bit format	*not available*	*not available*	0xF60	0xF63
PLC Control/Status	Boolean	*not available*	*not available*	0xF64	0xF65
PLC Error Code	int_16	*not available*	*not available*	0xF66	0xF67
CLC Life Counter	int_16	*not available*	*not available*	0xF68	0xF69
PLC Life Counter	int_16	*not available*	*not available*	0xF6A	0xF6B
Response Flag	Boolean	0xFFA	0xFFB	0xFFA	0xFFB
CLC_IRQ	Boolean	0xFFE	0xFFF	0xFFC	0xFFD
PC_IRQ	Boolean	0xFFC	0xFFD	0xFFE	0xFFF

Table A-1: Dual Port RAM Organization

Status and Control Bits

Note: For CLC-P02, the use of some Status and Control bits are dependent on the parametrization of C-0-0035 (PLC Communication option).

Status Bits

Bit 1: *Heartbeat* - bit toggles every two seconds, longer under some cases.

Bit 2: *Probe* - a 1 indicates CLC card has faulted into P_SOS OS. Card must be reset.

Bit 3: *Parameter Mode* - same as register 21 bit 1 (Register 21: System Status; 21-1: Parameter). A "1" indicates parameter mode is active.

Bit 4: *CLC Error* - same as register 21 bit 5 1 (Register 21: System Status; 21-5: Error. A "1" indicates an error has occurred on the CLC card. To resume operation, the error must be cleared by toggling bit 5 in register 1.

ID String 1

The CLC card writes the string “CLC-P” on power-up to identify the memory space usage. The CLC_DDE server looks for this string to validate the presence of the card.

CLC ID String

The CLC card writes its 16 character typecode on power-up to identify itself.

Master ID String

The device on the other side of the DPRAM writes its typecode to identify itself.

Time

The device on the other side of the DPRAM can write the current date and time.

32-bit Windows Time/Date Formats

Time

Bit Position:	0 1 2 3 4	5 6 7 8 9 A	B C D E F
Length:	5	6	5
Contents:	hours	minutes	2-second increments
Value Range:	0–23	0–59	0–29 in 2-second intervals

Date

Bit Position:	0 1 2 3 4 5 6	7 8 9 A	B C D E F
Length:	7	4	5
Contents:	year*	month	day
Value Range:	0–119	1–12	1–31

*(relative to 1980)

PLC control

Bit Position:	0	1	2
Length:	1	1	1
Contents:	run	error	time set

PLC Error Code

The PLC writes a 16-bit error code here when the error bit in PLC control is set. The CLC can optionally display this code.

CLC Life Counter

This 16-bit counter is incremented by the CLC every 4 milliseconds. The PLC or bus master may check this counter to determine if the CLC is still running.

PLC Life Counter

When the PLC option is enabled on the CLC, the PLC increments this counter every I-O scan. The CLC checks this counter every 200ms to determine if the PLC is running.

Response Flag

The CLC sets $FFB to 01 after responding to a communication request in the Comms Buffer. A PC task is responsible for clearing this after reading the response in the Comms Buffer.

CLC_IRQ

Any write to either byte by the PC triggers an interrupt on the CLC card.

Setting $FFF to 01 after entering a request in the Comms Buffer, the CLC will put the response in the Comms Buffer and set the Response Flag to 01.

Setting $FFF to 02 after entering a request in the Comms Buffer, the CLC will put the response in the Comms Buffer, set the Response Flag to 01, and write to PC_IRQ to generate an interrupt on the PC if enabled.

PC_IRQ

When the CLC writes to either bit, an interrupt is sent to the PC interrupt selected by jumpers S5, S6, S7 on the CLC-P01 and S1 DIP switches 5-8 on the CLC-P02. The CLC uses this for handshaking with the CLC DDE server.

A.4 Interrupts

The interrupts are used in conjunction with the DDE Server for improved communications and speed. This feature is only designed for NT 4.0 based systems.

The application program has no correlation to the hardware interrupts. They are only used within the serial interrupt routine for improved communications.

B VME (Versa Module Eurocard) Interface (CLC-V)

B.1 CLC-V System Memory Organization

The CLC is capable of accessing most of the full VME 4.3 gigabyte A32 extended address space; the CLC reserves memory above 0xF9FFFFFF. A configuration dialog box permits setup of each card's base address, and data and address access modes for up to 16 VME CLC cards on a single backplane. A typical 2 MB CLC at the default address is shown below.

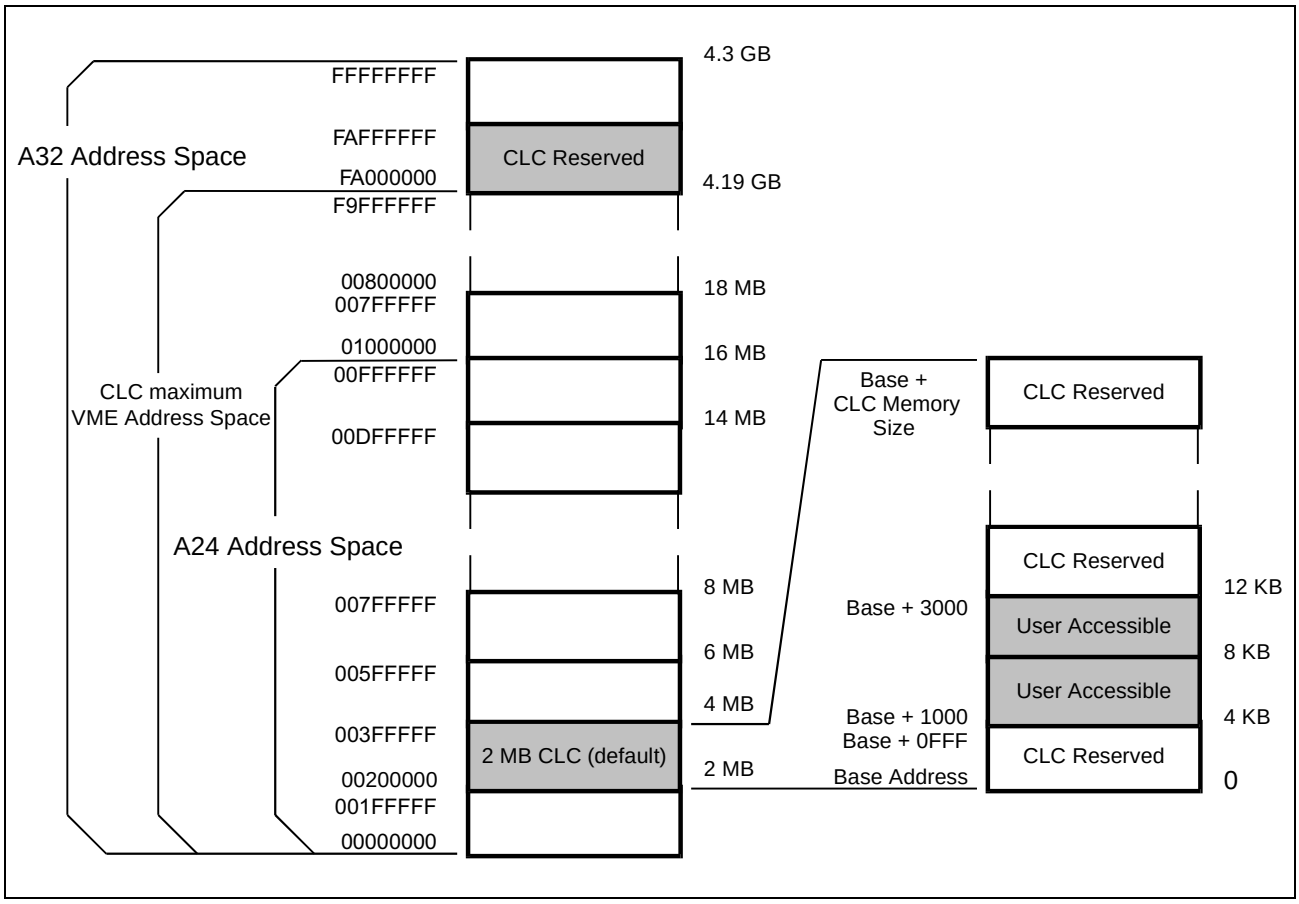


Figure B-1: CLC-V Memory Organization

Indramat supplies CLC cards in 2MB and 4MB versions. The base address of CLC cards may be configured to any integer (non-zero) multiple of the card's memory size within the VME A32 or A24 address space.

CLC-VME Memory Window

An 8 KB area of the CLC's VME memory is user accessible. This area contains the CLC's control and status registers, allowing direct user program access to the register contents. The area also provides a buffer for VME ASCII communications that can exchange messages between multiple VME cards on the backplane, and memory for both the CLC integer and floating point global variables.

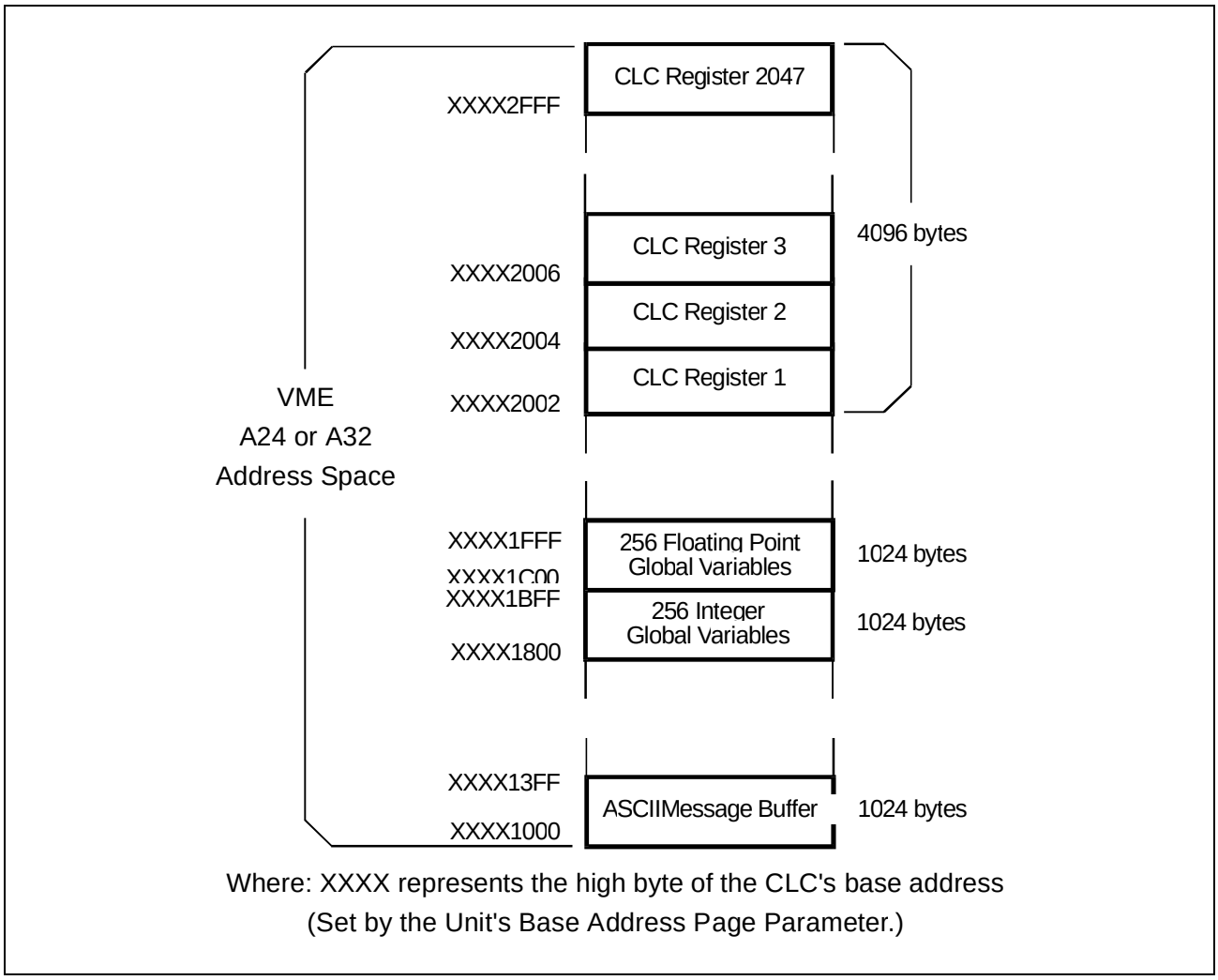


Figure B-2: VME Memory

Writing or reading CLC registers in the user accessible VME memory area directly affects I/O since the CLC's I/O driver software automatically exchanges the contents of the registers and I/O ports at different times during every 2 millisecond system cycle. The CLC registers and I/O ports are also subject to the logic strings specified in the CLC's I/O Mapper, and to any register forcing conditions in effect.

Warning

Uncontrolled Machine Motion

Changing VME memory locations can immediately start uncontrolled machine motion causing harm to personnel and machine.

⇒ Use caution and thoroughly understand all the potential effects of writing to CLC registers using VME memory access

ASCII messages may be sent and received across the VME backplane using a communication protocol and communication synchronization mailbox flags reserved in the VME short (A16) address space. Each CLC on the backplane allocates a 256 byte area of the short address space for its mailboxes. Each VME unit's mailbox-area base address may be configured on any exclusive 256 byte boundary.

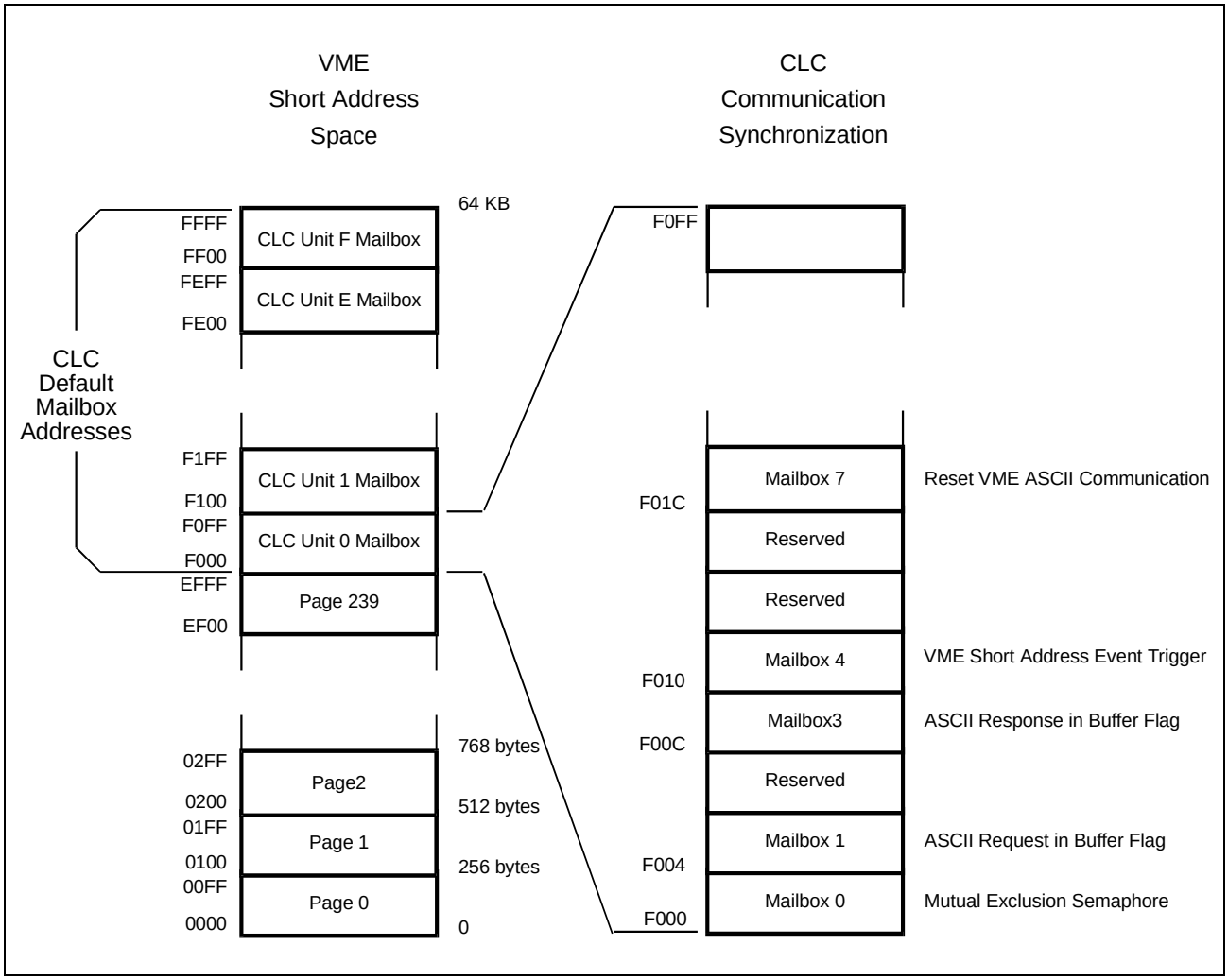


Figure B-3:Communication synchronization mailbox flags

Currently five mailbox addresses are specified. Reads and writes to the mailboxes must use the specified communication protocol. Any improper access can result in unpredictable results and system failure.

B.2 CLC-V Backplane Communication

The CLC is capable of exchanging data between multiple VME bus masters directly over the VME backplane. These data transfers use a 1k byte buffer in the standard (or extended) VME address space at offset 0x1000 from the base address of the CLC card. The format of messages sent to a CLC must conform to the specifications in the chapter on Direct ASCII Communication. A software handshaking protocol is required using mailboxes in the short VME address space to control the VME bus transfer. Mailbox operations are converted to VME read-modify-write cycles eliminating bus contention problems.

All CLC cards will propagate messages down the VME backplane. This permits a Host PC system to address and communicate with up to 16 CLC's in a VME card enclosure using a single serial connection to one CLC card.

The sequence of the communication handshake protocol is as follows:

1. The sending CLC reads mailbox 0 until bit 7 is 0. (E.g., loop on reading (or "poll") the mailbox until the bit is clear.) A 0 indicates that the sending (requesting) CLC is in control of the VME bus.

2. The sending CLC places the message in the VME message buffer.

3. The sending CLC then reads mailbox 1. Executing a read of the mailbox initiates the transfer operation. (Bit 7 should be 0, reading a 1 indicates a communication error.)

4. The sending CLC then reads mailbox3 (polls) until bit 7 is 0, indicating that a message has been received in response. (The receiving VME device is required to return a response message.)

5. The sending CLC writes to mailbox 0 releasing the ASCII communication channel. Any value may be written while the write operation itself terminates the transfer operation.

The CLC mailboxes are offset from the short VME address space base address as shown below:

- Mailbox 0 - read/write, offset 0x00
- Mailbox 1 - read only (do not write), offset 0x04
- Mailbox 3 - read only (do not write), offset 0x0C

CLC-V Direct Data Access

The CLC is capable of directly accessing shared VME memory. The Icon Language has a single VME icon providing data transfer in either direction, while the Text Language has separate VME/READ and VME/WRITE commands. Either language may be used to transfer data between VME memory and the CLC local memory. Data sets that may be transferred include system registers, integer or floating point variables, or absolute or relative point table entries. One or more of the elements of a data set may be transferred with a single command.

The direct access mode also may be used to obtain or modify the contents of the CLC registers. CLC registers are located offset 0x2000 from the CLC card's base address, in the VME standard (or extended) memory address space.

VME Memory Access Limitations

Since the VME bus is capable of accessing memory using three data widths (D8, D16 and D32), there are limitations on the data transfer modes that may be used to exchange data between the CLC tables and registers, and VME memory. Attempting to transfer I32, 4 byte VME data to a CLC I16, 2 byte integer variable target, creates a problem. A D32 transfer operation cannot be used, because the highest 2 bytes of the 4 byte operation would overwrite the next two memory locations. Therefore, only a D16 or D8 transfer mode can be used, and the transfer is limited to VME write only.

A similar problem can result from trying to write from a smaller source (say U8) to a wider VME target (say I32). Since the source is a single byte, only a D8 transfer operation would be allowed. Transferring one byte leaves the upper bytes undefined; therefore, the I32 target data would be undefined. Consequently, transfers from a source smaller than a target are limited to VME read only.

The CLC's byte swapping support of both the Motorola and Intel byte order on the VME bus adds other considerations for signed and unsigned integer formats.

Supported CLC - VME bus Data Transfer Operations				
Source Data	**Destination Data**	**D32 capability**	**D16 capability**	**D8 capability**
I32	I32	read/write	read/write	read/write
	I16		write only	write only
	I8			write only
	U32	write only	write only	write only
	U16	read only	read/write	read/write
	U8			write only
	F32	read/write	read/write	read/write
I16	I32		read only	read only
	U16		read only	read only
	F32		read only	read only
I8	I32			read only
	U16			read only
	F32			read only
U32	I32	read only	read only	read only
	U16	read only	read only	read only
	F32	read only	read only	read only
U16	I32	write only	read/write	read/write
	I16		write only	write only
	I8			write only
	U32	write only	read/write	read/write
	U16		write only	write only
	U8			write only
	F32	write only	read/write	read/write
U8	I32			read only
	U16			read only
	F32			read only
F32	I32	read/write	read/write	read/write
	I16		write only	write only
	I8			write only
	U32	write only	write only	write only
	U16		read/write	read/write
	U8			write only
	F32	read/write	read/write	read/write
P28 (point table)	P28	read/write	read/write	read/write

Figure B-4: VME bus Data Transfer Operations

B.3 VME Configuration

The VME Configuration includes the specification the VME system's memory addresses, data modes, and system control signals. CLC VME configuration beyond the standard default configuration is only required if one of the following conditions is valid for the VME system:

- There will be more than one CLC VME motion control card in the VME card cage.
- There will be more than one VME master (i.e., a CLC motion control with a VME PLC).
- There are VME I/O cards that must occupy the default CLC memory space.
- The VME system designer requires use of the CLC default memory space.

There are three ways to alter the CLC VME configuration:

- by using the VME Configuration dialog box from the Setup menu
- by directly modifying VME parameters through the Overview window also accessed from the Setup menu
- by Direct ASCII Communication through the serial port.

To save a VME configuration the CLC system must be in parameter mode.

VME Address and Access Modes

Installing a CLC into a VME system requires specifying the VME unit number that the CLC card will respond to, the VME address spaces that the card will occupy, and the manner in which other devices will access the CLC memory (the access modes). The CLC is configured by using the VME Configure dialog box to specify the following values.

Unit Number

The Unit number identifies the card within the VME card cage. The unit number is selected by the CLC's front panel 16 position MODE switch. The unit number is a logical identifier, independent of the physical card position.

Short Address Page

The Short address page sets the base address of a 256 byte (A16) address space. The value specified provides the high byte of the 16-bit address and the lower byte is zeroed. Within the short address space the CLC provides 8 mailboxes that are used for synchronization.

Address Range

The Address range specifies the width of the address bus used to access the CLC's memory from the VME bus. The CLC permits A32 extended VME addressing, (32-bit) and A24 standard VME addressing (24-bit) when addressing accessible RAM.

Data Width

The Data Width determines the number of bytes transferred with a single memory access operation. The CLC VME permits VME bus access in D8 (8-bit, single byte), D16 (16-bit word, double byte), and D32 (32-bit longword, or 4 byte) data widths.

Base Address Page

The Base address page sets the base address of the block of contiguous CLC VME memory within the VME system memory space. The CLC

VME memory block cannot have a base address of less than the memory size of the CLC card. The lowest base address the CLC can use depends upon the size of the CLC memory; it must be a multiple of the CLC memory size. A 2 Meg CLC card can occupy from 2MB to 4MB, a 4 Meg CLC can occupy 4MB to 8MB, etc.

Note: The base address should never be set lower than the address space required by the largest VME card in the VME system.

The value specified by the Base Address Page provides the high 8 bits (A24 addressing) or 16 bits (A32 addressing) of the address, the lower 16 bits are zeroed (64k pages). The base address must be an integer multiple of the CLC card's RAM size. Specific areas in the CLC's memory are accessed using offsets from this specified base address.

VME System Signals

A user must specify a number of settings that determine how and if the CLC is to generate and/or respond to VME bus control signals. These include selection of the CLC as the slot 1 master providing VME bus arbitration, VME bus release modes, and the CLC's responses to certain VME bus error conditions.

Bus Arbitration

Fair Arbitration Enabled

Fair Arbitration Enabled resolves simultaneous same-priority bus requests (on the BR0 - BR3 bus request lines) from multiple VME cards. Activation of this option requires that the other VME cards (requesters) use the FAIR protocol for releasing and requesting the bus request lines. The CLC arbiter can then service each requester at the same priority level sequentially, in the order of the requesting card's physical position in the VME bus request daisy-chain.

If the FAIR request option is enabled, the CLC's VME bus controlling gate array will not request control of the VME bus until the bus request line BRx* of the VME bus is released by all requesters. This is the start of a new arbitration round. Each requester who has sampled the bus request line high is now allowed to request mastership of the VME bus.

After the request line has been asserted again, all requesters in the current round are defined. Bus mastership will be given to the requester in order of priority, starting at the most prioritized location in the daisy-chain, which is closest to the bus arbiter. When this new master gains control of the bus it releases its bus request line, while the other requesters remain asserted.

When the master has finished with the bus (BBSY goes inactive) the next arbitration round begins, without the previous master, since the FAIR request option prevents the last master from asserting its bus request line until BRx* has been released by all other requesters. This guarantees that the low prioritized masters can obtain the bus.

The CLC's gate array recognizes a negated bus request when it is high for a minimum of 20 nanoseconds. In order that all participants are able to sample the high state of the request signal line, the gate array asserts its request output not earlier than 50 nanoseconds after it has detected the BRx* signal high.

Bus Request Level

Bus Request Level indicates the VME bus request hardware priority that the CLC has when requesting the VME bus. Four bus request levels are available: BR0 (lowest) to BR3 (highest). The bus request level must be set manually using switches SW5-6 and SW5-7 on the CLC card.

VME Slot 1 Mode

VME slot 1 mode provides a pop-down list of three bus arbitration modes. These selections are only active if the CLC card is the VME system "Slot 1" controller (arbiter). (Note that the VME bus's BR0 - BR3 daisy-chains also determine the priority of bus requesters according to the physical positioning on the VME backplane.)

Prioritized - arbitrates the VME bus according to hardware bus requests using the four daisy-chained bus request lines: BR0, BR1, BR2 and BR3. BR3 has the highest priority, BR0 the lowest. A higher priority bus request releases the bus to the higher priority requester when the current bus access cycle is completed.

Round-Robin - ignores priority among the hardware request lines (BR0 - BR3), and all request lines have the same level of priority. When the bus is released, the CLC scans each lower numbered bus request line. If no request is found the scan continues to the next higher priority, continuously scanning from BR0 through BR3.

Prioritized Round-Robin - Bus request BR3 has priority over BR2, BR1 and BR0. BR2, BR1 and BR0 are arbitrated using the round-robin algorithm. This mode should only be used when there will be only one master with priority BR3 requesting the bus. Prioritized Round Robin is the CLC default.

Bus Release Modes

Release Every Cycle

Release Every Cycle releases the VME bus after each bus transfer is done. This mode permits another VME card to request the bus after every bus cycle. Since the bus is re-arbitrated after every cycle this mode reduces available bus response-time.

Release On Bus Clear

Release On Bus Clear permits the release of the bus if requested by another bus master. If no request is pending, the CLC retains control of the bus. Release on bus clear overrides the timing limitation set by ROR.

ROR Inhibit Time

ROR Inhibit Time (Release On Request) provides pop-down list selection of a hold-off time before the bus is released to a pending request from another bus master. Time may be selected from 0.5 microsecond to 64 microseconds. ROR ensures that the CLC remains master of the bus for the selected minimum amount of time. ROR is overridden by checking Release On Bus Clear.

Bus Interface

VME Bus Access Error Fatal

VME bus access error fatal, if checked, generates a fatal error if the CLC attempts to access the VME bus and fails to receive a bus acknowledge signal. The time before the error occurs is set by the Access Timeout selected.

SYSFAIL Signal Fatal

SYSFAIL signal fatal, if checked, generates a fatal error if a failure is received from the VME bus. (A failure is indicated by a message in the CLC server and VisualMotion Toolkit System selection from the Status menu for all system errors.)

External RMW Cycle Allowed

External RMW cycle allowed, if checked, permits **R**ead/**M**odify/**W**rite access cycles by other VME cards in the system. RMW reserves the bus to the master generating the RMW, locking out the CLC until the external master releases the bus. When invoked RMW has the highest priority of any normal bus activity and cannot be interrupted.

Access Timeout

Access Timeout provides a pop-down list permitting selection of the time interval before a failed VME bus acknowledge generates a fatal error. Time may be set from 16 to 64,000 microseconds.

VME Bus Parameters

Note: The VME parameters are 200 series system parameters.

Three CLC parameters determine how the CLC interacts with the VME bus system.

Parameter 200, Bus Arbitration Mode, selects the method of bus arbitration if the CLC is the VME bus arbiter.

Parameter 201, Bus Release Mode, determines the timing and bus release mode when the CLC is the VME bus master.

Parameter 202, VME/CLC Interface, determines how bus related errors are handled, bus access timing, and enables/disables external read/modify/write (RMW) cycles.

For a description of the options available for CLC control of the VME bus, see the section on VME System Signals in this chapter. For details on the control bits for the above parameters, refer to the 200 series parameters in Chapter 3, Parameters, of the VisualMotion 6.0 Reference Manual.

VME Address and Access Mode Parameters

Setting the VME addressing and data width requires three parameters for each unit. The parameter numbers are related to the VME unit number as offsets from the base unit #0 parameters (210, 211, and 212), according to the expression:

Unit #n (1 - 15) parameter number = Unit #0 parameter number + (n * 5)

The first parameter is a single byte specifying the high eight bits of the Short Address Page 16-bit address.

The second parameter is a single byte specifying the VME access modes. (VME standard 24-bit or extended 32-bit address width; and 8-bit, 16-bit or 32-bit data width.)

The third parameter is a 16-bit word specifying the CLC's VME Window Address Page (e.g., the CLC card's base address for standard and extended VME memory).

For example:

Parameter 210 sets the high eight bits of the Short Address Page for unit #0.

Parameter 211 sets the VME address and data width that unit #0 uses to access memory.

Parameter 212 specifies the base address for a CLC set to unit #0.

For further information, refer to Chapter 3, Parameters, of the VisualMotion 6.0 Reference Manual.

C Non-CLC Hardware

C.1 VME Card Cage

Indramat currently supplies a seven-position VME card cage as an option. The cage provides a VME backplane, integral power supply, card cooling fans, a front-accessible memory backup battery, cutouts for four DB-25 connectors, and termination for AC input power.

VME Card Cage Secondary Battery

User programs stored in the 2MB SRAM use a 3.5 volt, "AA" size, lithium cell with pigtail leads for battery backup. The battery holder, located at the bottom left of the card cage front panel, does not have electrical contacts, it is only a mechanical holder. The battery must be electrically connected to the VME backplane by connecting the positive (red) lead of the battery to the "+5V" terminal of terminal strip X5, and the negative (blue) lead to the "0V" terminal.

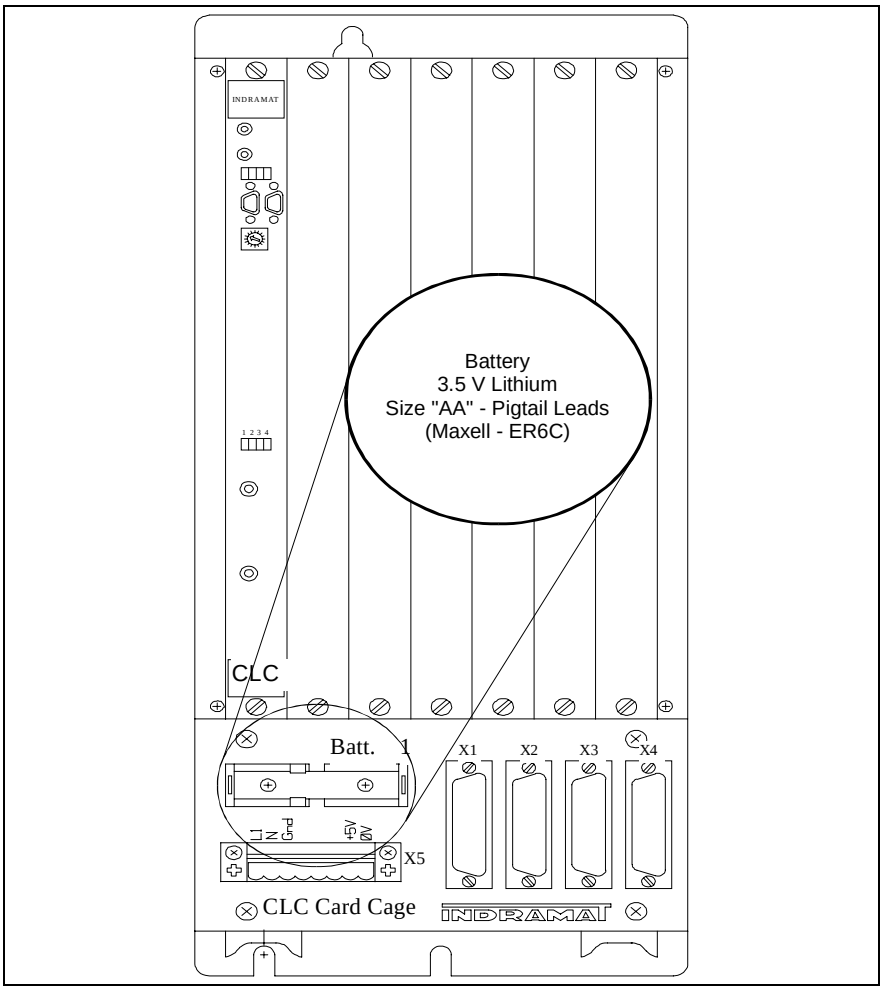


Figure C-1: VME Card Cage Secondary Battery

Ordering a Replacement Battery for the VME Card Cage

To order a replacement backup battery for the VME card cage, specify part number **226423**. The replacement battery you will receive has a blue three-terminal connector. This connector needs to be removed before placing the leads into the two-terminal Phoenix connector (part number **219628**) supplied with the card cage. A 1 Kohm resistor (part number **210336**) has been added in line with the Red wire which connects to the +5V (pin 1). The blue wire connects to the 0V (pin 2).

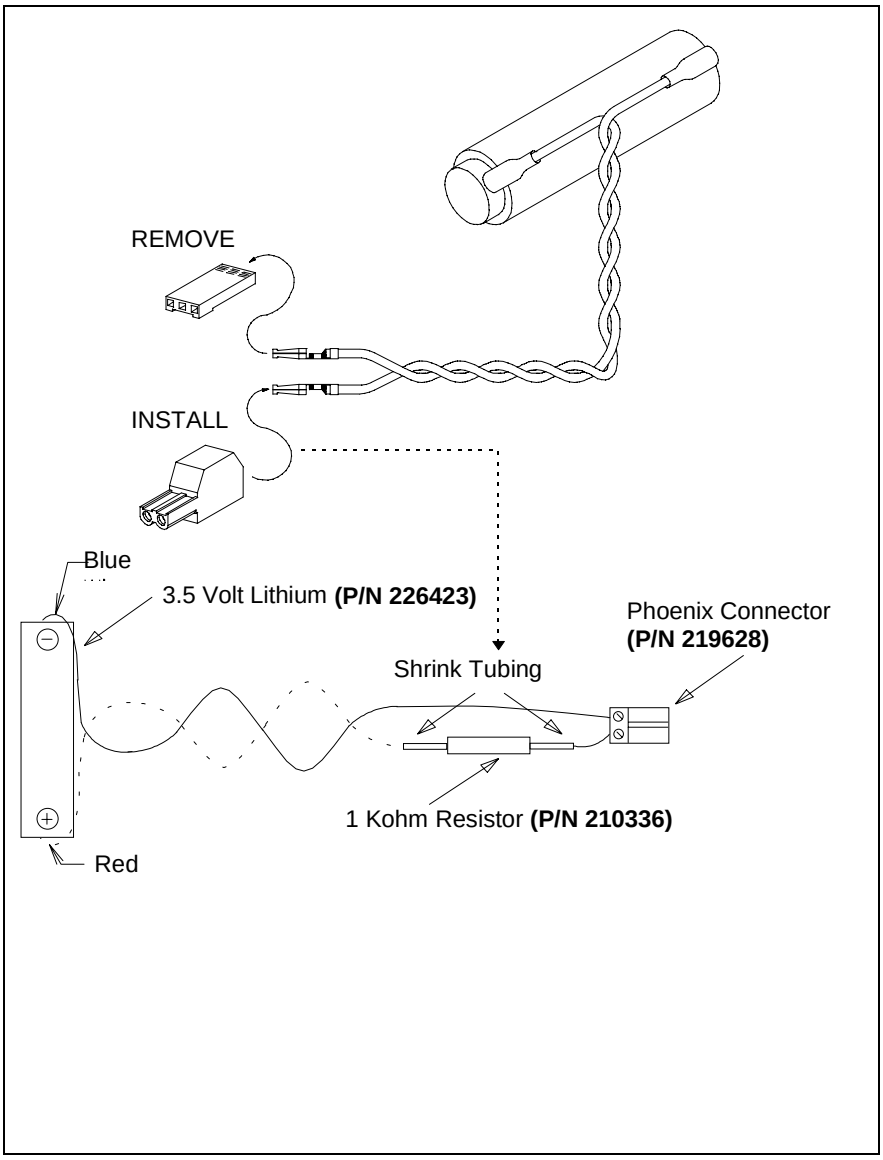


Figure C-2: Replacement Battery for the VME Card Cage

Caution

CLC-V SRAM memory loss

Loss of SRAM memory will result if STDBY line battery backup is not existent.

⇒ Before installing the CLC-V card into a third party card cage that provides battery backup, check the card cage manual to ensure that the card cage uses the VME STDBY line for backup and that both positions of SW6 are OFF.

VME Backplane Jumpers

The present version of the VME card cage (VMCC) have auto-bus-grant connectors which eliminate the need to manually jumper unused slots between cards. Prior to VMCC Version 1.2, five jumpers must be installed in each unused slot, as shown below.

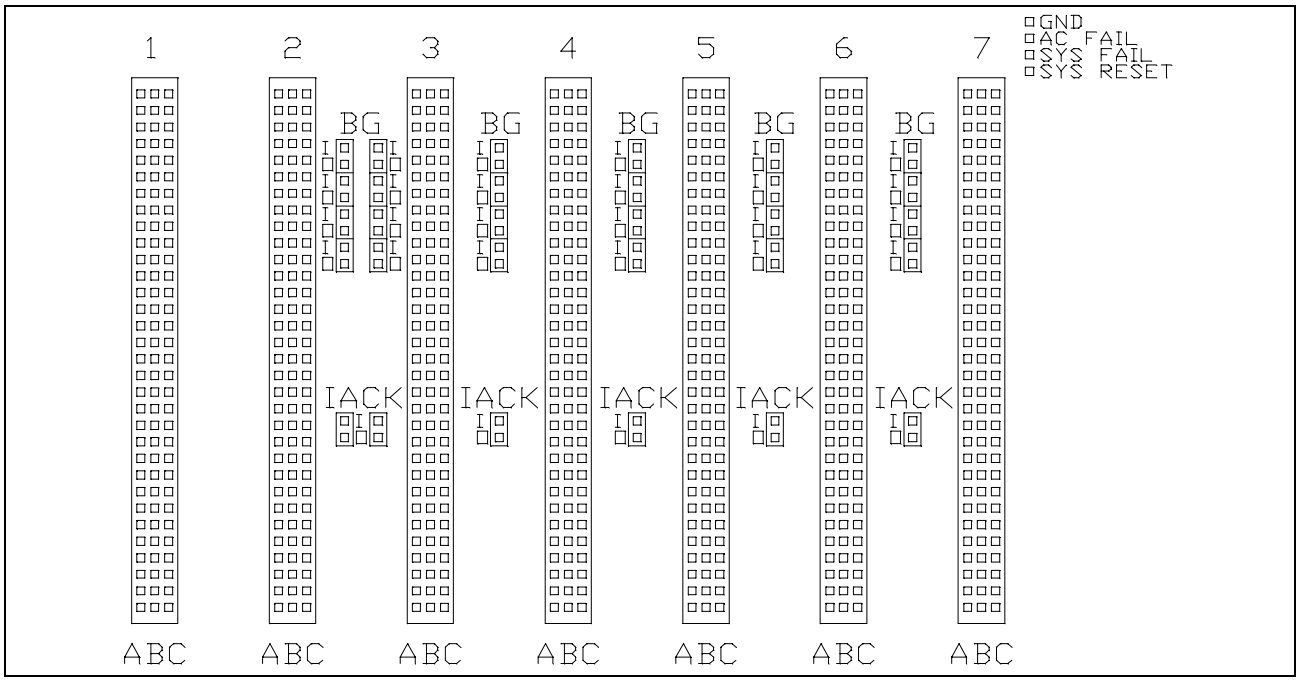


Figure C-3: Replacement Battery for the VME Card Cage

Connecting AC Input Power

Use only three conductor power to connect to the terminal strip (X5) at the lower left front of the card cage.

1. Connect the "hot" AC supply lead to X5, terminal "L1."
2. Connect the "neutral" AC supply lead to X5, terminal "N."
3. Connect the AC "ground" lead to X5, terminal "G."

C.2 CLC VME I/O Systems

Using the open architecture of the VME bus, the CLC-V is capable of using any compatible VME bus I/O card for system input/output. However, development of appropriate driver software is involved. Writing an I/O driver requires complete knowledge of all parts of the VME bus system and the characteristics of your program under all operating conditions. Due to the difficulty of writing and integrating I/O drivers, Indramat provides drivers for a number of I/O cards and subsystems.

Indramat currently supplies integrated software drivers for the following digital I/O cards:

- Xycom XVME-201
- Xycom XVME 202 PAMUX
- Xycom XVME 244
- Pentland MPV922

Indramat offers the following analog I/O cards for which no drivers are supplied:

- Xycom XVME 500
- Xycom XVME 505

Note: Jumpers and/or switches on I/O cards must be properly configured for operation. Be sure to check the information under the appropriate heading in this section and the manufacturers installation instructions.

In addition to the I/O card configuration, the CLC parameters listed below are used to provide the CLC with information about the I/O system configuration. Both parameters and I/O board configuration must be consistent.

CLC Card parameter	Parallel I/O function
C-0-0005	I/O card type descriptor
C-0-0006	I/O card setup information
C-0-0007	I/O line direction
C-0-0008	I/O card VME base address

DEA I/O (DIAX Drive-Resident I/O)

Each Indramat DDS digital drive may optionally include up to three parallel I/O cards, each providing 15 inputs and 16 outputs. The three DEA cards are designated: DEA 4.2M, DEA 5.2M, and DEA 6.2M. Each card is internally configured; therefore, the cards are independent of position in the drive.

The card data is accessed using the SERCOS cyclic telegram providing high-speed distributed I/O on up to 8 drives. DEA I/O card data is accessed through CLC I/O registers 40 through 87, and are sequentially mapped to the I/O cards for drives 1 through 8 in ascending order.

CLC Register	Register function	Register Label Name
40	Drive 1 - DEA 4.2 inputs	DDS1_1IN
41	Drive 1 - DEA 4.2 outputs	DDS1-1OUT
42	Drive 1 - DEA 5.2 inputs	DDS1_2IN
43	Drive 1 - DEA 5.2 outputs	DDS1_2OUT
44	Drive 1 - DEA 6.2 inputs	DDS1_3IN
45	Drive 1 - DEA 6.2 outputs	DDS1_3OUT
. . .	. . .	. . .
86	Drive 8 - DEA 6.2 inputs	DDS 8_ 8IN
87	Drive 8 - DEA 6.2 outputs	DDS 8_ 8OUT

Each DEA card must be enabled by setting an appropriate CLC parameter. Set the parameter to a logic "1" to enable the card.

DDS card	CLC Axis parameter
DEA 4.2M	A-0-0011
DEA 5.2M	A-0-0014
DEA 6.2M	A-0-0017

If the parameter is set to "0", the I/O data are not included in the SERCOS telegram and the corresponding CLC registers are not updated. The DEA I/O data may still be read through the SERCOS service channel using the appropriate SERCOS parameter.

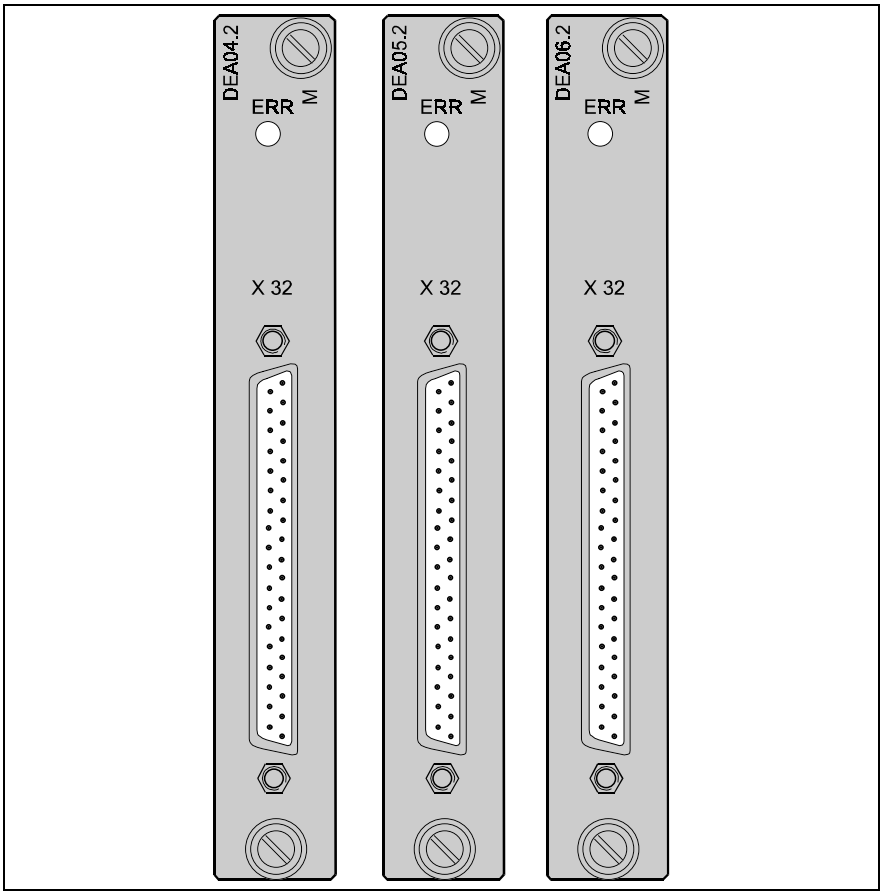


Figure C-4: DEA Front Panel (4.2M, 5.2M, 6.2M)

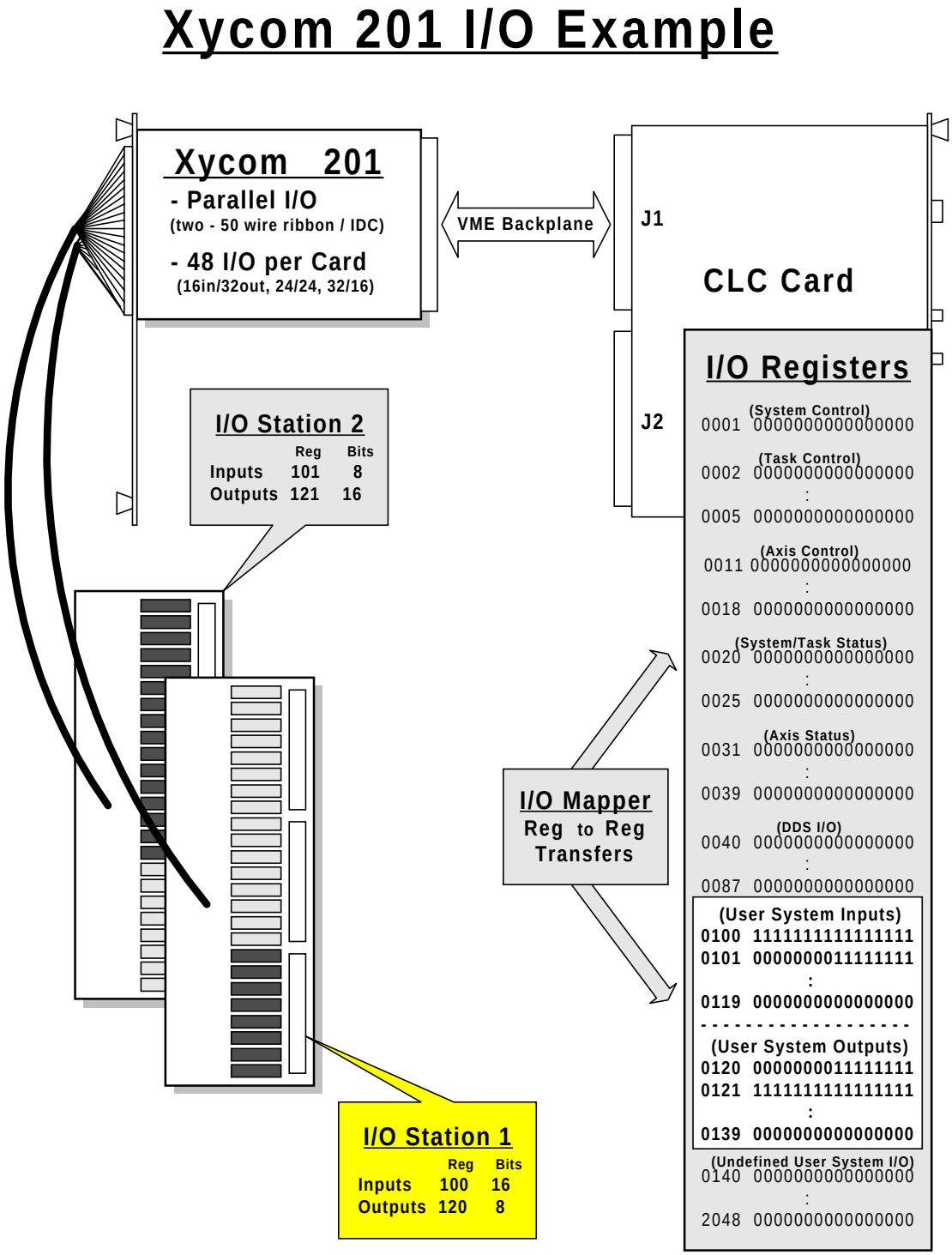
The DEA I/O card front panel uses a 37-pin D-sub connector. Connector pinouts are listed in the table below.

DEA pin	function	DEA pin	function
1	IN 0	16	OUT 0
2	IN 1	17	OUT 1
3	IN 2	18	OUT 2
4	IN 3	19	OUT 3
5	IN 4	20	OUT 4
6	IN 5	21	OUT 5
7	IN 6	22	OUT 6
8	IN 7	23	OUT 7
9	IN 8	24	OUT 8
10	IN 9	25	OUT 9
11	IN 10	26	OUT 10
12	IN 11	27	OUT 11
13	IN 12	28	OUT 12
14	IN 13	29	OUT 13
15	IN 14	30	OUT 14
		31	OUT 15
35	0 Vext (return)	37	~ +24 Vext (source)
32	not used	34	not used
33	not used	36	not used

DEA Electrical Specifications

Specification	minimum	typical	maximum
Vext (user supply)	+18V	+24V	+32V
Iext (user supply)	0.15A	0.2A	2.2A
Each input			
I/O input (high)	+12V	+24V	+32V
I/O input (low)	0	<1V	+3V
Each output			
high	+10V	+18V	+32V
low	0	+0.8V	+1V

Xycom XVME-201 Digital I/O



Each Xycom XVME-201 digital module provides a maximum of 48 I/O points as two groups of 24. A ribbon cable connects each 24-point group to an Opto-22 station that contains appropriate Opto-22 modules. Up to eight XVME-201 cards can be accommodated, allowing a maximum of 384 I/O points.

Each card occupies 1K of the short VME address space. Card base addresses are assigned on 1K boundaries within the 64K short address space by properly placing jumpers on the card and setting CLC parameter C1.8.

The XVME-201 is a half-height 3U VME card. If the XVME-201 you receive has the standard half-height front panel, installation of the card into the full-height VME card enclosure requires replacement of the half-height front panel with a full-height front panel available from Xycom. This replacement is required for proper forced air cooling of cards in the VME enclosure.

For most systems, the Indramat supplied driver for the Xycom XVME-201 card is all that is required to use the card as the interface to an Opto-22 I/O system.

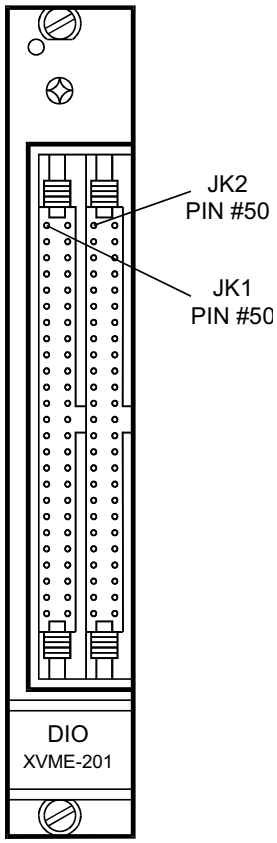
Accessing I/O Lines

If programming the card through the VME bus interface is desired, a thorough understanding of the XVME-201 card and the Motorola MC68230 PI/T chip used to manage the cards' I/O lines is required. Please refer to the Xycom and Motorola documentation for the necessary information.

The CLC I/O registers, 100 through 115, are reserved as input registers for the eight XVME-201 cards. The outputs from the card are similarly mapped to CLC output registers 120 through 135. The even numbered CLC registers correspond to the XVME-201 JK1 card connector. The odd numbered registers correspond to the JK2 card connector.

Specification	minimum	typical	maximum
Vext (user supply)	+18V	+24V	+32V
Iext (user supply)	0.15A	0.2A	2.2A
Each input			
I/O input (high)	+12V	+24V	+32V
I/O input (low)	0	<1V	+3V
Each output			
high	+10V	+18V	+32V
low	0	+0.8V	+1V

XVME-201 Card # - Connector	CLC input register number	CLC output register number
1 - JK1	100	120
1 - JK2	101	121
2 - JK1	102	122
2 - JK2	103	123
3 - JK1	104	124
3 - JK2	105	125
4 - JK1	106	126
4 - JK2	107	127
5 - JK1	108	128
5 - JK2	109	129
6 - JK1	110	130
6 - JK2	111	131
7 - JK1	112	132
7 - JK2	113	133
8 - JK1	114	134
8 - JK2	115	135



I/O Configuration Parameters

Parameter C1.5, Parallel I/O Device, must be set to "3", indicating that the Xycom XVME-201 is the I/O device type.

Parameter C1.6, Parallel I/O Device Setup, must be set to individually enable each of the sixteen possible Opto-22 stations that are installed. Enable each station by setting the corresponding bit in the parameter to a logical "1", according to the mapping below.

0000000000000000

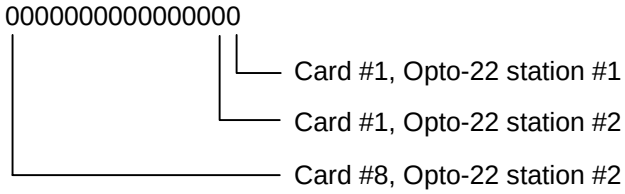
- Card #1, connector JK1 (CLC register 100)
- Card #1, connector JK2 (CLC register 101)
- Card #8, connector JK2 (CLC register 135)

Example:

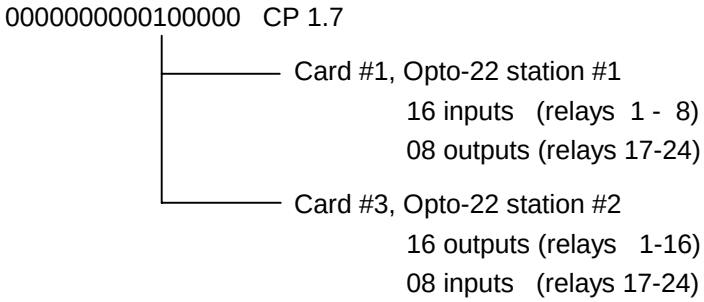
0000000000110011 CP 1.6

- Card #1, both stations enabled
- Card #3, both stations enabled

Parameter C1.7, Parallel Device, I/O Device Direction, specifies the combination of inputs and outputs that will be available on each Opto-22 station.



Example:


0000000000100000 CP 1.7
Card #1, Opto-22 station #1
16 inputs (relays 1 - 8)
08 outputs (relays 17-24)
Card #3, Opto-22 station #2
16 outputs (relays 1-16)
08 inputs (relays 17-24)

The table below shows how the least significant bit (LSB) in parameter C1.7 (XVME-201 card #1, Opto-22 station #1) affects the number of card inputs and outputs, and the mapping to CLC registers. Sixteen position Opto-22 stations only connect to the first 16 module positions, position 17 through 24 are empty.

CLC Parameter C1.7	Opto-22 relays	Opto-22 relay function	CLC Inputs register - bit	CLC Outputs register - bit
LSB = 0 (default)	1 - 8	Input	100 - 1 to 8	
	9 - 16	Input	100 - 9 to 16	
	17 - 24	Output		120 - 1 to 8
LSB = 1	1 - 8	Output		120 - 1 to 8
	9 - 16	Output		120 - 9 to 16
	17 - 24	Input	100 - 1 to 8	

Parameter C1.8, Parallel Device VME Start Address, must be set to specify the VME bus base address for the card. The parameter is set as a 16-bit hexadecimal word equal to the desired base address and must be equal to the address set by the XVME-201 on-board jumpers, JA10 through JA15. (See the jumper settings and corresponding addresses in the section below.)

XVME-201 Card Configuration

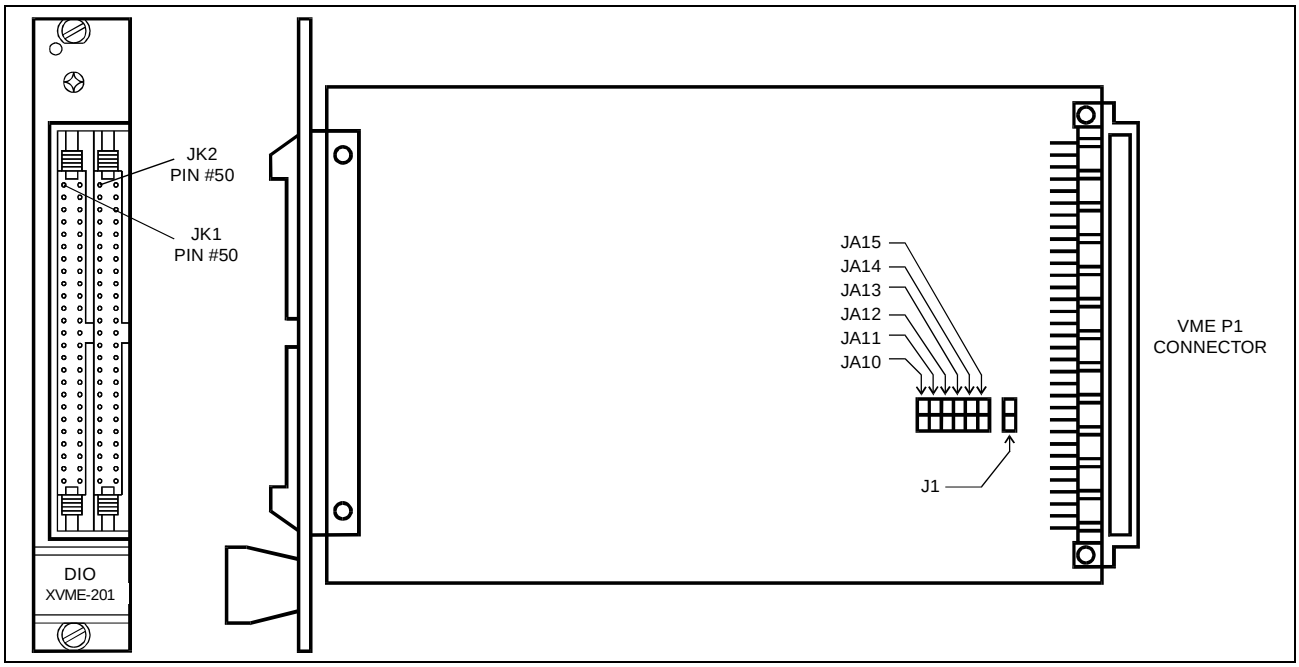


Figure C-5: XVME-201 Card - Jumper Locations

Jumper J1 determines the VME bus short I/O Address Modifier Codes that can affect the card.

XVME-201 Jumper	open/closed	function
J1	closed	card responds to 0x2D VME AM code (Supervisory only)
J1 (default)	open	card responds to 0x2D and 0x29 VME AM codes (Supervisory and non-privileged)

Jumpers JA10 through JA15 set the base address of the card within the VME 64k Short I/O Address space on 1k boundaries. The table below lists the base addresses and corresponding jumpers for XVME-201 cards one through eight, if parameter C1.8 is 0x0000.

Xycom XVME-201		Address Jumper					
Card #	Base Address	JA10	JA11	JA12	JA13	JA14	JA15
1	0x0000	closed	closed	closed	closed	closed	closed
2	0x0400	open	closed	closed	closed	closed	closed
3	0x0800	closed	open	closed	closed	closed	closed
4	0x0C00	open	open	closed	closed	closed	closed
5	0x1000	closed	closed	open	closed	closed	closed
6	0x1400	open	closed	open	closed	closed	closed
7	0x1800	closed	open	open	closed	closed	closed
8	0x1C00	open	open	open	closed	closed	closed

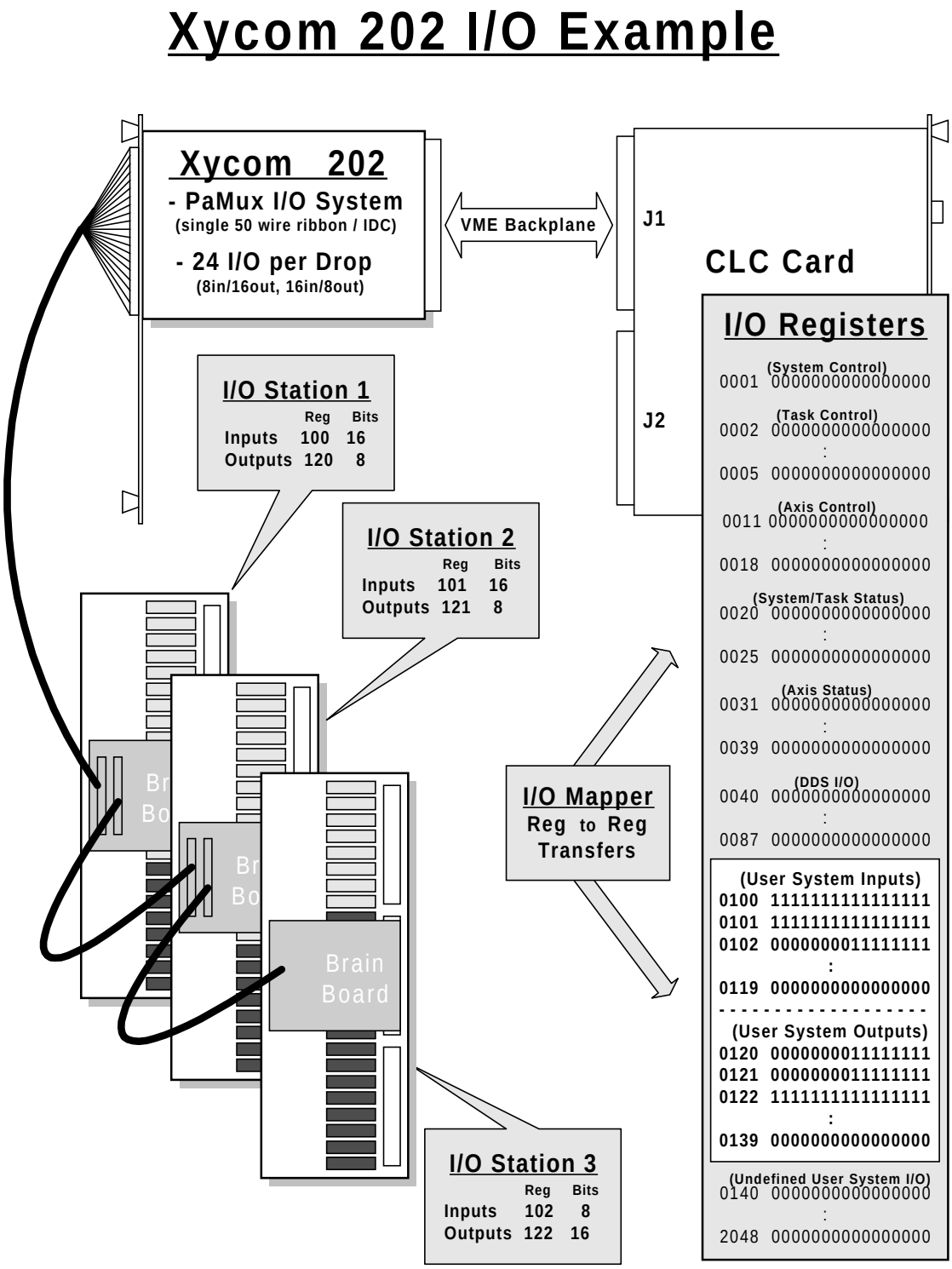
The base address of each card may be calculated using the expression:

*Short Address Base Address = [(XVME-202 card #-1) * 0x400] + Parameter C1.8*

Xycom XVME-201 Characteristics

Characteristic	Specification
Number of Channels	48
Power Requirements	+5 Vdc, 1.3A typ., 1.5A max
Dimensions	VME single height (3U) 150mm x 116.7mm
Temperature Operating Non-Operating	 0 to 65 C (32 to 149 F) -40 to 85 C (-40 to 158 F)
Humidity	5 to 95% RH, non-condensing
VMEbus Compliance	• Complies with VMEbus Standard Rev. C.1 • A15:D8(0) DTB Slave • Size - Single • Base address jumper-selectable on 1K boundaries within the VMEbus short I/O address space

Xycom XVME-202 (PAMUX™ I/O)



The Xycom XVME-202 is a single height (3U) VME card designed to be used as an interface between the VME bus and a PAMUX distributed I/O subsystem. A single 50 conductor ribbon cable from the XVME-202 is daisy-chained to each PAMUX station with a maximum extension of 500 feet.

One card can provide distributed parallel I/O for up to 16 PAMUX stations, each with up to 32 configurable I/O points. A total of 512 I/O may be controlled using a single XVME-202 card.

Each card occupied 1K of the short VME address space. Card base addresses are assigned on 1K boundaries within the 64K short address space by properly placing jumpers on the card and setting CLC parameter C1.8.

The XVME-202 is a half-height 3U VME card. If the XVME-202 you receive has the standard half-height front panel, installation of the card into the full-height VME card enclosure requires replacement of the half-height front panel with a full-height front panel available from Xycom (part number XVME-941). This is replacement is required for proper forced air cooling of cards in the VME enclosure.

Indramat's I/O drivers for the XVME-202 map CLC input registers 100 through 115 to inputs on each of the 16 possible PAMUX stations. Registers 120 through 135 are mapped to the outputs on the PAMUX units. If a station with less than 32 I/O bits is used, the number of inputs and outputs are selected by setting Parameter C1.7.

PAMUX station base address	CLC register		PAMUX station base address	CLC register	
	input	output		input	output
0	100	120	8	108	128
1	101	121	9	109	129
2	102	122	10	110	130
3	103	123	11	111	131
4	104	124	12	112	132
5	105	125	13	113	133
6	106	126	14	114	134
7	107	127	15	115	135

I/O Configuration Parameters

Parameter C1.5, Parallel I/O Device, must be set to "1", indicating that the Xycom XVME-202 is the I/O device type.

Parameter C1.6, Parallel I/O Device Setup, must be set to individually enable each of the sixteen PAUMUX I/O stations that are installed.

Example:

0000000000000101 CP 1.6

PAMUX station addresses 0 and 2 enabled

Parameter C1.7, Parallel I/O Device Direction, must be set to enable the individual

0000000000000000
- Card #1, Opto-22 station #1
- Card #1, Opto-22 station #2
- Card #8, Opto-22 station #2

Example:

0000000000100000 CP 1.7
- Card #1, Opto-22 station #1
  - 16 inputs (relays 1 - 8)
  - 08 outputs (relays 17-24)
- Card #3, Opto-22 station #2
  - 16 outputs (relays 1-16)
  - 08 inputs (relays 17-24)

Parameter C1.8, Parallel I/O Device I/O Direction, must be set to specify the base address for the card. The parameter is set as a 16-bit hexadecimal word equal to the desired base address and must be equal to the address set by the XVME-202 on-board jumpers, JA10 through JA15. (See the jumper settings and corresponding addresses in the section below.)

XVME-202 Card Configuration

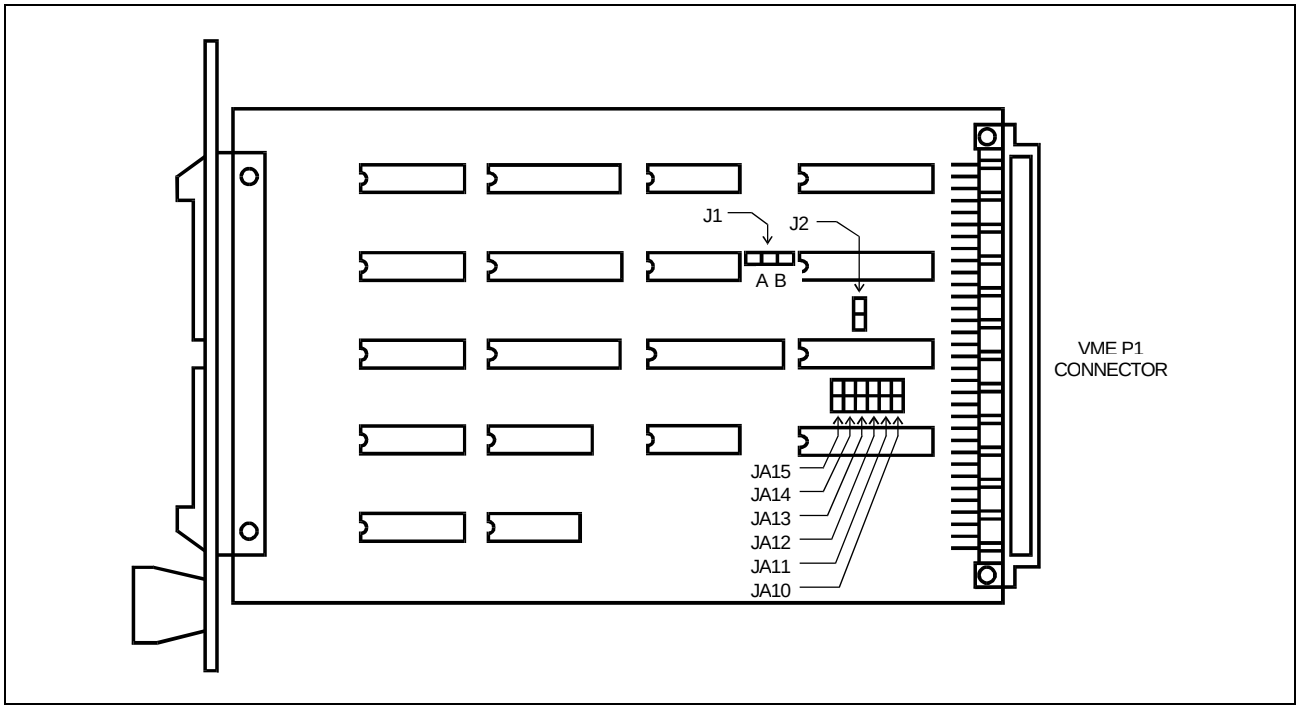


Figure C-6: XVME-202 Card - Jumper Locations

Jumper J1 selects between a board mounted oscillator and the VME clock. Jumper J2 determines the VME bus short I/O Address Modifier Codes that can affect the card.

XVME-201 Jumper	default	function
J1	B	A - selects the optional on-board oscillator. B - selects VME bus SYSCLK
J2	CLOSED	OPEN - card responds to 0x2D VME AM code (Supervisory only) CLOSED - card responds to 0x2D and 0x29 VME AM codes (Supervisory and non-privileged)

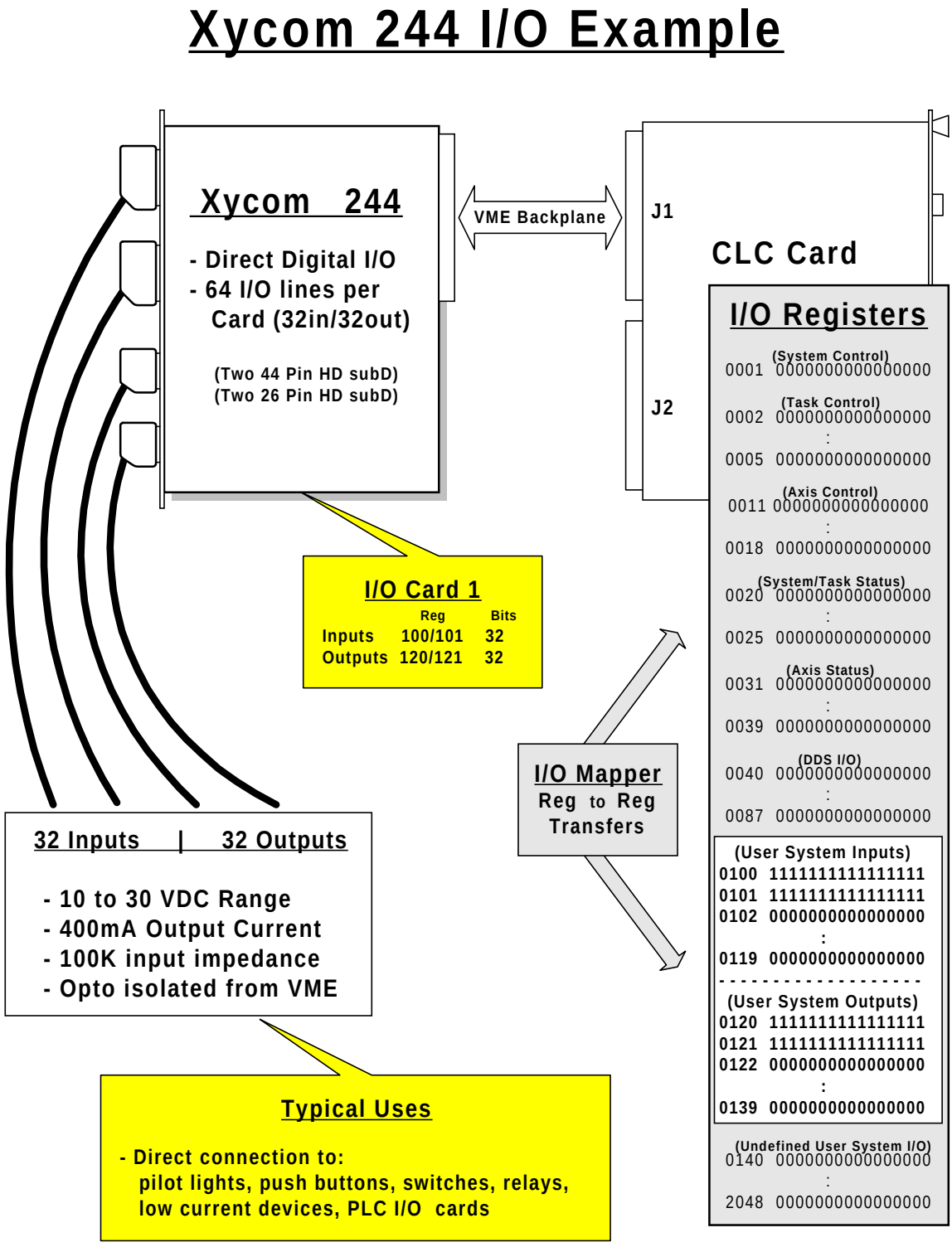
Jumpers JA10 through JA15 set the base address of the card within the VME 64k Short I/O Address space on 1k boundaries. The table below lists the base addresses and corresponding jumpers for XVME-201 cards one through eight.

Xycom XVME-201		Address Jumper					
Card #	Base Address	JA10	JA11	JA12	JA13	JA14	JA15
1	0x0000	closed	closed	closed	closed	closed	closed
2	0x0400	open	closed	closed	closed	closed	closed
3	0x0800	closed	open	closed	closed	closed	closed
4	0x0C00	open	open	closed	closed	closed	closed
5	0x1000	closed	closed	open	closed	closed	closed
6	0x1400	open	closed	open	closed	closed	closed
7	0x1800	closed	open	open	closed	closed	closed
8	0x1C00	open	open	open	closed	closed	closed

The base address of each card may be calculated using the expression:

*Short Address Base Address = (XVME-201 card #-1) * 0x400*

Xycom XVME-244 Digital I/O

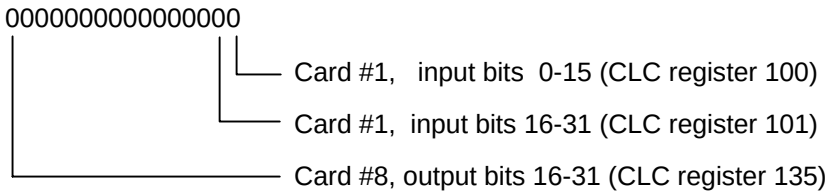


The Xycom XVME-244 digital module contains 32 optically-isolated inputs and 32 optically-isolated outputs directly mounted on the card. Each card is configured as two stations of 16-inputs and two stations of 16-outputs. The CLC's XVME-244 I/O drivers accommodate up to eight cards, providing a maximum of 256 inputs and 256 outputs.

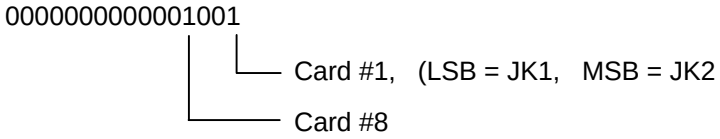
I/O Configuration Parameters

Parameter C1.5, Parallel I/O Device, must be set to "5", indicating that the Xycom XVME-244 is the I/O device type.

Parameter C1.6, Parallel I/O Device Setup, must be set to enable the installed XVME-244 cards. Each card is enabled by setting a corresponding bit in parameter C1.6 to '1'.


0000000000000000
Card #1, input bits 0-15 (CLC register 100)
Card #1, input bits 16-31 (CLC register 101)
Card #8, output bits 16-31 (CLC register 135)

CLC parameter C1.7, Parallel I/O Device Direction, sets an on-board filtering option for the XVME-244 inputs by setting the bits corresponding to the appropriate card. The direction of the inputs is fixed at 32 in and 32 out for each board.



Parameter C1.8, Parallel Device VME Start Address, must be set to specify the VME bus base address for the card. The parameter is set as a 16-bit hexadecimal word equal to the desired base address and must be equal to the address set in SW1 positions 1 through 6 (see the SW6 table in the board configuration section below).

XVME-244 Board Configuration

The XVME-244 card requires on-board jumper configuration for the range of operating voltage and VME bus address and supervisory control signals.

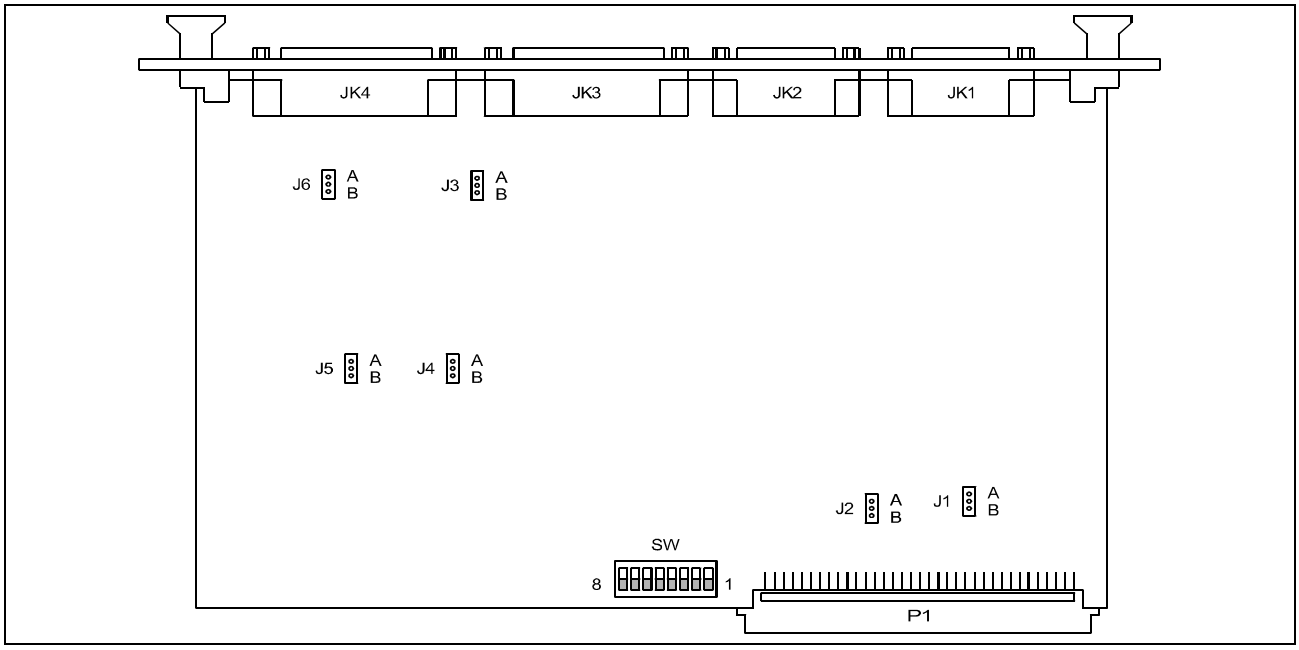


Figure C-7: XVME-244 Card - Jumper Locations

Configuration Jumpers

The jumper settings are application-dependent. The following are the default settings:

Jumper	Default	Description
J1	B	A - clear output registers on VME SYSFAIL signal B - do not clear outputs on SYSFAIL
J2	B	A - assert VME SYSFAIL when FAIL LED on B - do not assert SYSFAIL when FAIL LED on
J3	A	Group 1 A - power supply range 15 - 30VDc B - power supply range 10 - 15 VDc
J4	A	Group 2 A - power supply range 15 - 30VDc B - power supply range 10 - 15 VDc
J5	A	Group 3 A - power supply range 15 - 30VDc B - power supply range 10 - 15 VDc
J6	A	Group 4 A - power supply range 15 - 30VDc B - power supply range 10 - 15 VDc

Configuration Switch SW1

Switch SW1 positions 1 through 6 set the base address of the XVME-244 card. Switch positions 7 and 8 must be set to the indicated defaults.

SW1 position	Default setting	Function
1	Open (see address table)	Open - card responds to VME bus A10 = "1" Closed - card responds to VME bus A10 = "0"
2	Open (see address table)	Open - card responds to VME bus A11 = "1" Closed - card responds to VME bus A11 = "0"
3	Open (see address table)	Open - card responds to VME bus A12 = "1" Closed - card responds to VME bus A12 = "0"
4	Open (see address table)	Open - card responds to VME bus A13 = "1" Closed - card responds to VME bus A13 = "0"
5	Open (see address table)	Open - card responds to VME bus A14 = "1" Closed - card responds to VME bus A14 = "0"
6	Open (see address table)	Open - card responds to VME bus A15 = "1" Closed - card responds to VME bus A15 = "0"
7	Open	Open - supervisor mode Closed - supervisor and non-private mode
8	Open	Open - VME standard (A24) address space Closed - VME short address space

XVME-244 Base Address

Xycom XVME-244		SW1 position					
Card #	Base Address	1	2	3	4	5	6
1	0x0000	closed	closed	closed	closed	closed	closed
2	0x0400	open	closed	closed	closed	closed	closed
3	0x0800	closed	open	closed	closed	closed	closed
4	0x0C00	open	open	closed	closed	closed	closed
5	0x1000	closed	closed	open	closed	closed	closed
6	0x1400	open	closed	open	closed	closed	closed
7	0x1800	closed	open	open	closed	closed	closed
8	0x1C00	open	open	open	closed	closed	closed

The base address may be calculated by the formula:

*base address = [(card # -1) * 0x400] + C1.8*

Note: The card's base address must also be set in CLC parameter C1.8.

Note: For all cards, if C1.8 = 0x2000,
- the first card is at 0x2000,
- the second card is at 0x2400,
  etc.

Western Reserve SmartMux (PAMUX™)

"Information To Be Added"

Pentland MPV922 VME Digital I/O

The Pentland MPV922 card provides 40 inputs and 32 outputs. The input bits are mapped to CLC registers 140 through 142, and the output bits are mapped to CLC registers 145 through 146.

The card must be configured to respond to the 0x29 and 0x2D VME AM codes.

C.3 PC/ISA I/O Systems

The CLC-P cannot directly access PC-compatible I/O cards based on the Industry Standard Architecture (ISA) bus. A commercially available soft PLC host with it's own custom control software (along with a library of drivers) is needed to access I/O cards since the CLC-P does not directly control the ISA bus. This PLC host writes any required I/O back to the CLC-P.

D Example Programs

D.1 Example 1: Sequencer Application

The following sample program contains a Sequencer that integrates the use of Subroutines, Branches and an I/O device.

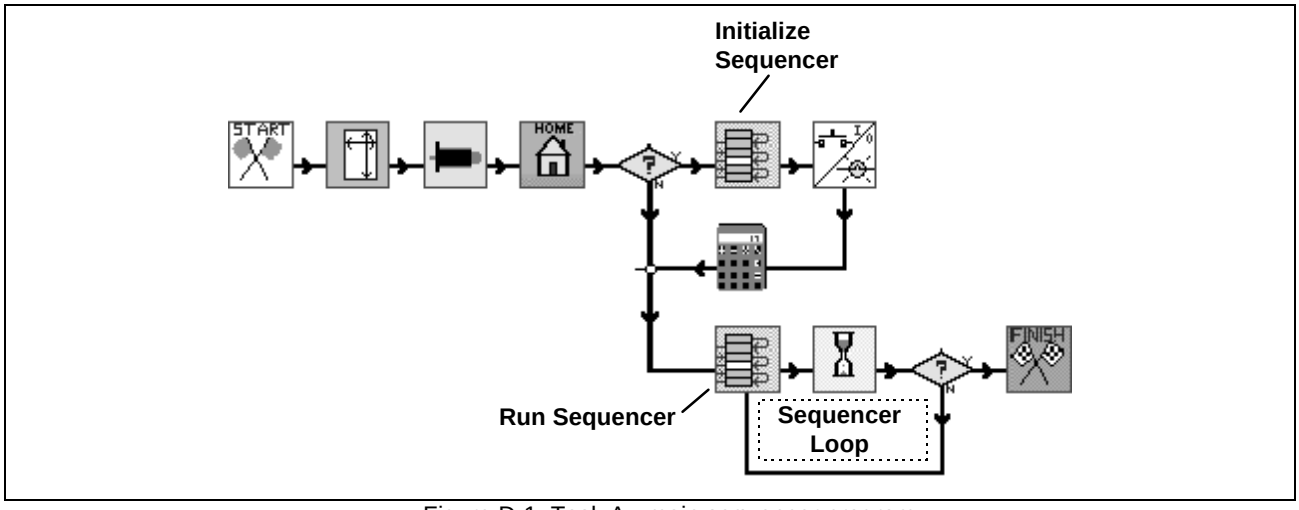


Figure D-1: Task A - main sequencer program

The first time this program is executed, the sequencer is *initialized.* The I/O state of a predefined register (reg. 100) is changed to indicate that the initialization has been completed. The program flow will then reset a cycle counter and Run the sequencer. The program will continue running in a sequencer loop until a WAIT condition or control register bit is triggered.

When the program is stop and restarted, the I/O state of the predefined register will cause the program to branch to the ***Run Sequencer*** icon and continue into the sequencer loop.

A Sequencer editor can be used to make changes with the program running. If the sequencer is *initialized* again, any of these changes will be overwritten with the initialization data. To prevent this from happening, the first branch in the program checks the I/O state of register 100 (initialization indicator). If the Sequencer has already been initialized, the program will only *run* the Sequencer. This maintains any current Sequencer changes. To initialize the Sequencer again, the predefined I/O register needs to be changed to its original state.

On-line changes can be made to the value of any predefined function argument within a Sequencer list, as long as all the steps are in their original order. If the original order is changed, or if a new function or step list is added, the program execution needs to be stopped before any changes can be saved to the CLC card. When the program is restarted the Sequencer will then execute the current step list/s.

To create this program: Follow the *Program Instructions* while referring to the *Program Layout* figures for dialog box input information. Refer to the *Program Icons* section and the CLC Reference manual for icon descriptions and any general programming information.

Program Instructions

Before the main sequencer program is created, all corresponding subroutines must first be created. A sequencer is a series of subroutines that are called up in a particular order to produce a process.

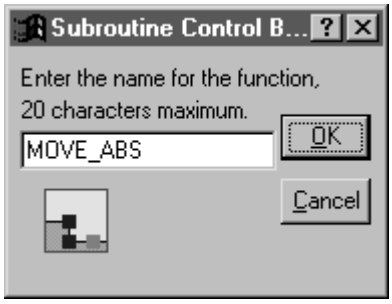
Using VisualMotion toolkit's icon programming environment, create the following four subroutines:

Subroutine Name	Usage
MOVE_ABS	Moves the programmed axis to a programmed absolute distance.
SET_IO	Sets an I/O bit on and off to emulate the function of a gripper.
COUNT	Cycle counts in increments of one for every complete cycle of the sequencer.
WAIT	Creates a delay in milliseconds after each absolute move.

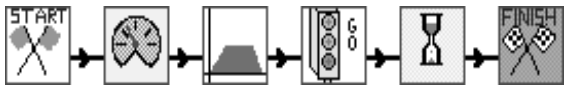
MOVE_ABS subroutine

1. Select ***Edit ⇒ Add Subroutine*** to create the **MOVE_ABS** subroutine.

2. Name the subroutine **MOVE_ABS** and click on *OK*. This will open a new programming window with a Start and a Finish icon.



3. Create the subroutine illustrated below according to the selections and values given in their corresponding dialog boxes. Refer to Program Layout - Function / Subroutine - MOVE_ABS, Figure D-2.

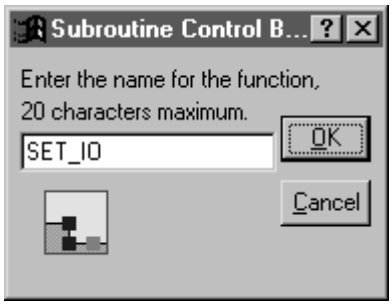


4. When done, return to *Task A* by selecting ***View ⇒ Task A***.

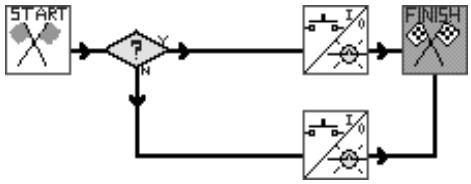
SET_IO subroutine

1. Select ***Edit ⇒ Add Subroutine*** to create the **SET_IO** subroutine.

2. Name the subroutine **SET_IO** and click on *OK*. This will open a new programming window with a Start and a Finish icon.



3. Create the subroutine illustrated below according to the selections and values given in their corresponding dialog boxes. Refer to Program Layout - Function / Subroutine - SET_IO, Figure D-3.

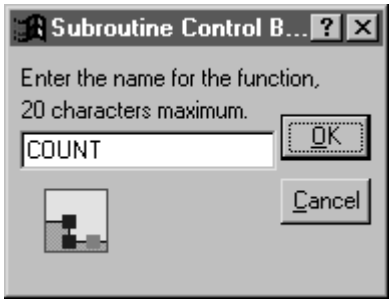


When done, return to *Task A* by selecting ***View ⇒ Task A***.

COUNT subroutine

1. Select ***Edit ⇒ Add Subroutine*** to create the **COUNT** subroutine.

2. Name the subroutine **COUNT** and click on *OK*. This will open a new programming window with a Start and a Finish icon.



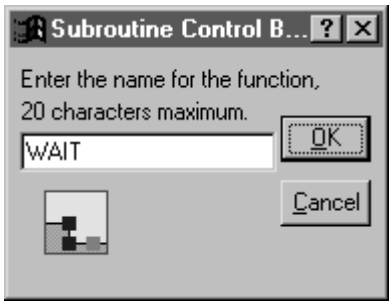
3. Create the subroutine illustrated below according to the selections and values given in their corresponding dialog boxes. Refer to Program Layout - Function / Subroutine - COUNT, Figure D-4.

When done, return to *Task A* by selecting ***View ⇒ Task A***.

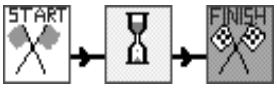
WAIT subroutine

1. Select ***Edit ⇒ Add Subroutine*** to create the **WAIT** subroutine.

2. Name the subroutine **WAIT** and click on *OK*. This will open a new programming window with a Start and a Finish icon.



3. Create the subroutine illustrated below according to the selections and values given in their corresponding dialog boxes. Refer to Program Layout - Function / Subroutine - WAIT, Figure D-5.



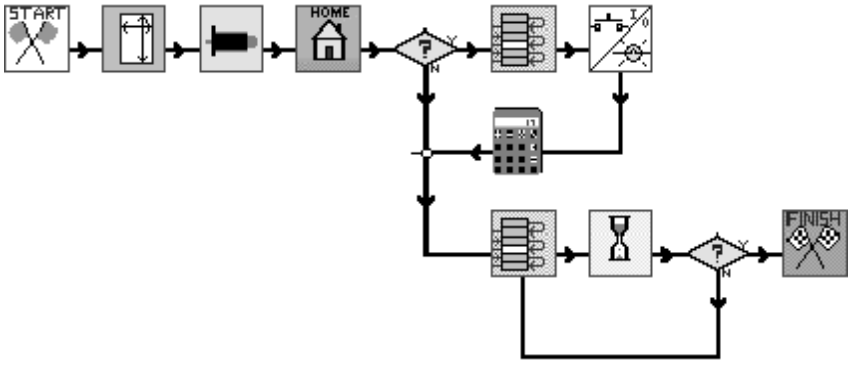
When done, return to *Task A* by selecting ***View ⇒ Task A***.

Main Sequencer Program

Task A

After all four subroutines are created, the next step is to create the main sequencer program. This main sequencer program will be used to initiate and run, in a predefined order or list, the previous four (4) subroutines.

1. Create the subroutine illustrated below according to the selections and values given in their corresponding dialog boxes.



Program Layout

Function / Subroutine - MOVE_ABS

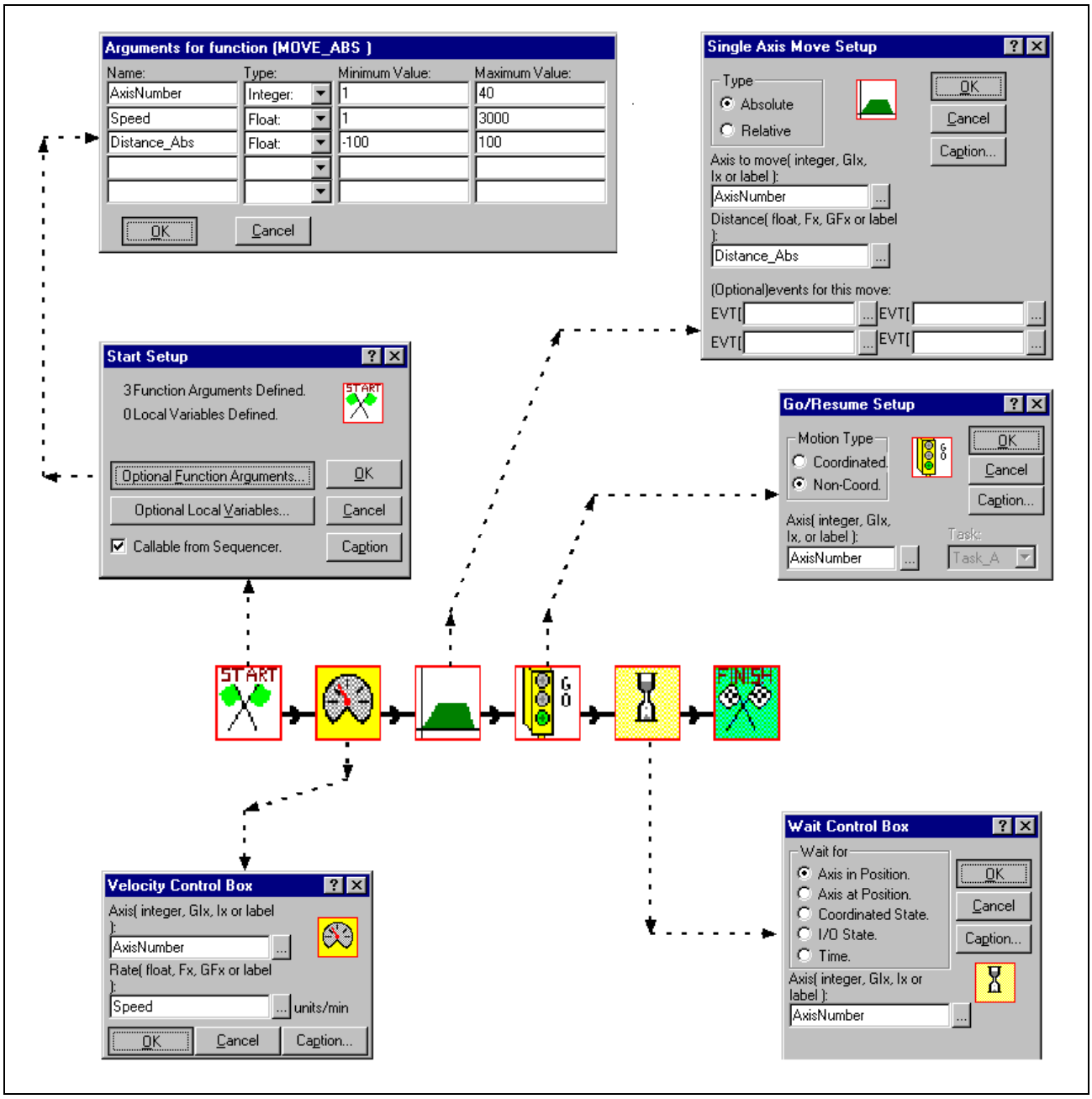


Figure D-2: MOVE_ABS subroutine

Function / Subroutine - SET_IO

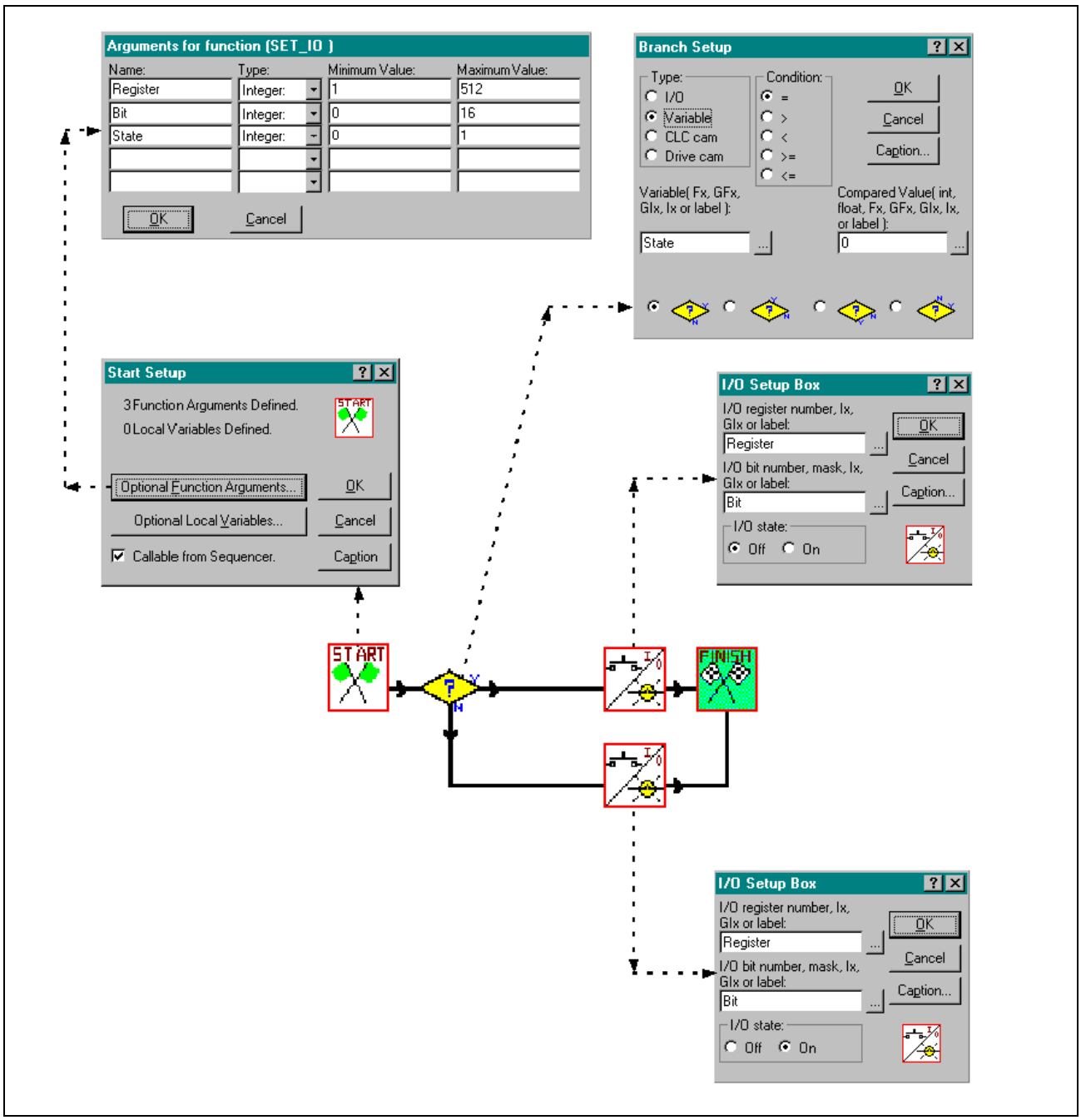


Figure D-3: SET_IO subroutine

Function / Subroutine - COUNT

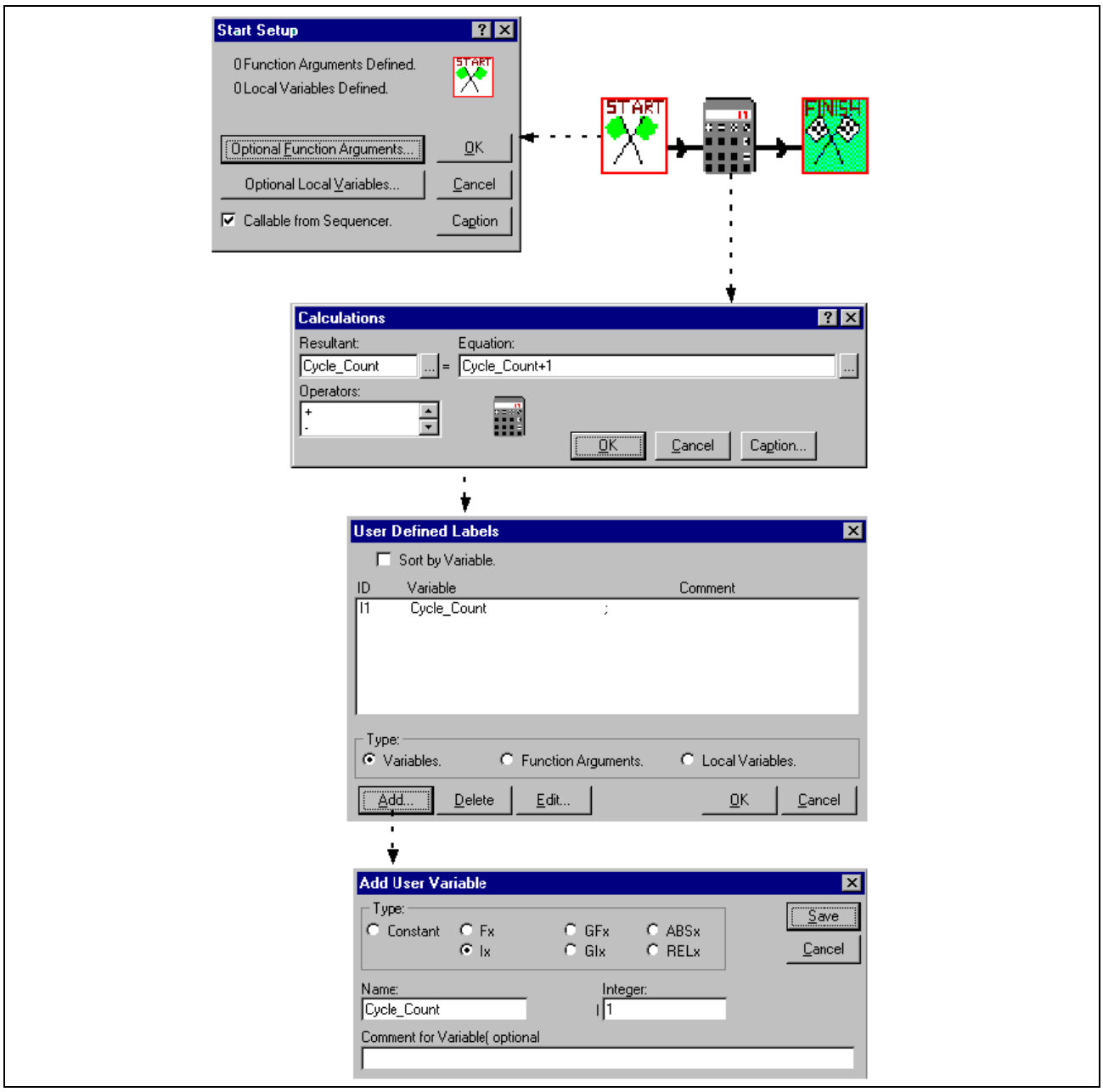


Figure D-4: COUNT subroutine

Function / Subroutine - WAIT

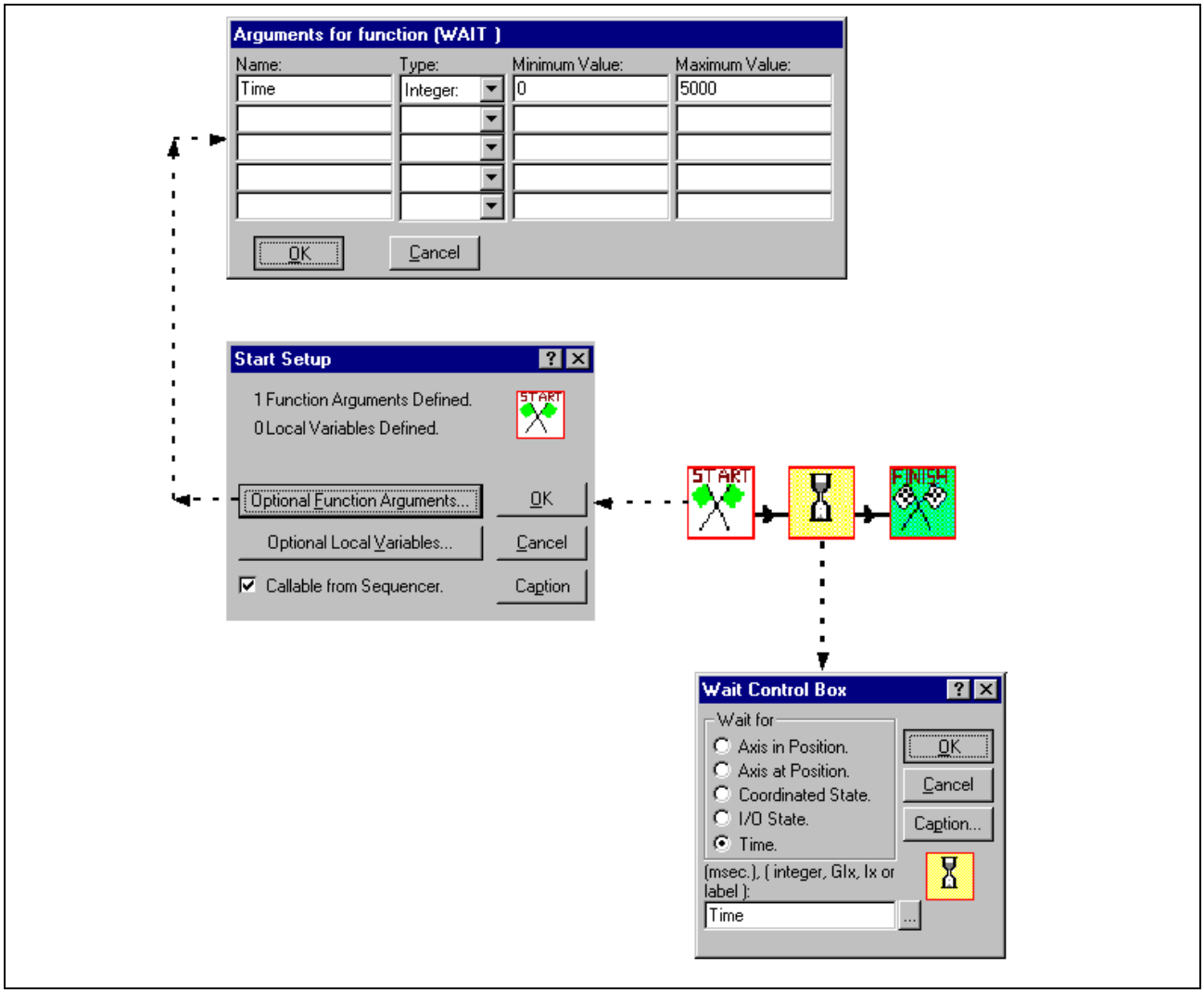


Figure D-5: WAIT subroutine

Task A, Main Sequencer - Part 1

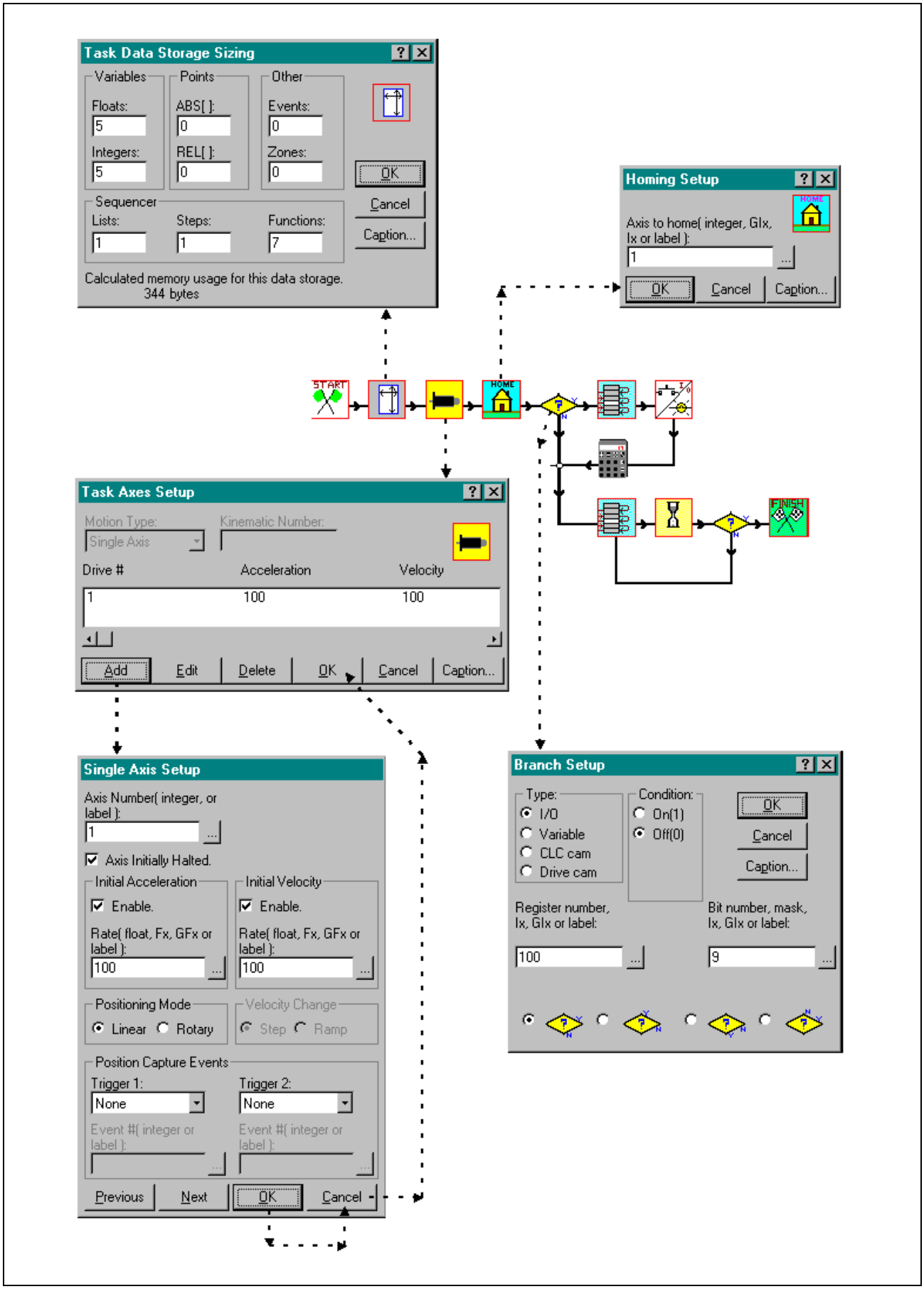


Figure D-6: Main Sequencer - Part 1 of 3

Task A, Main Sequencer - Part 2

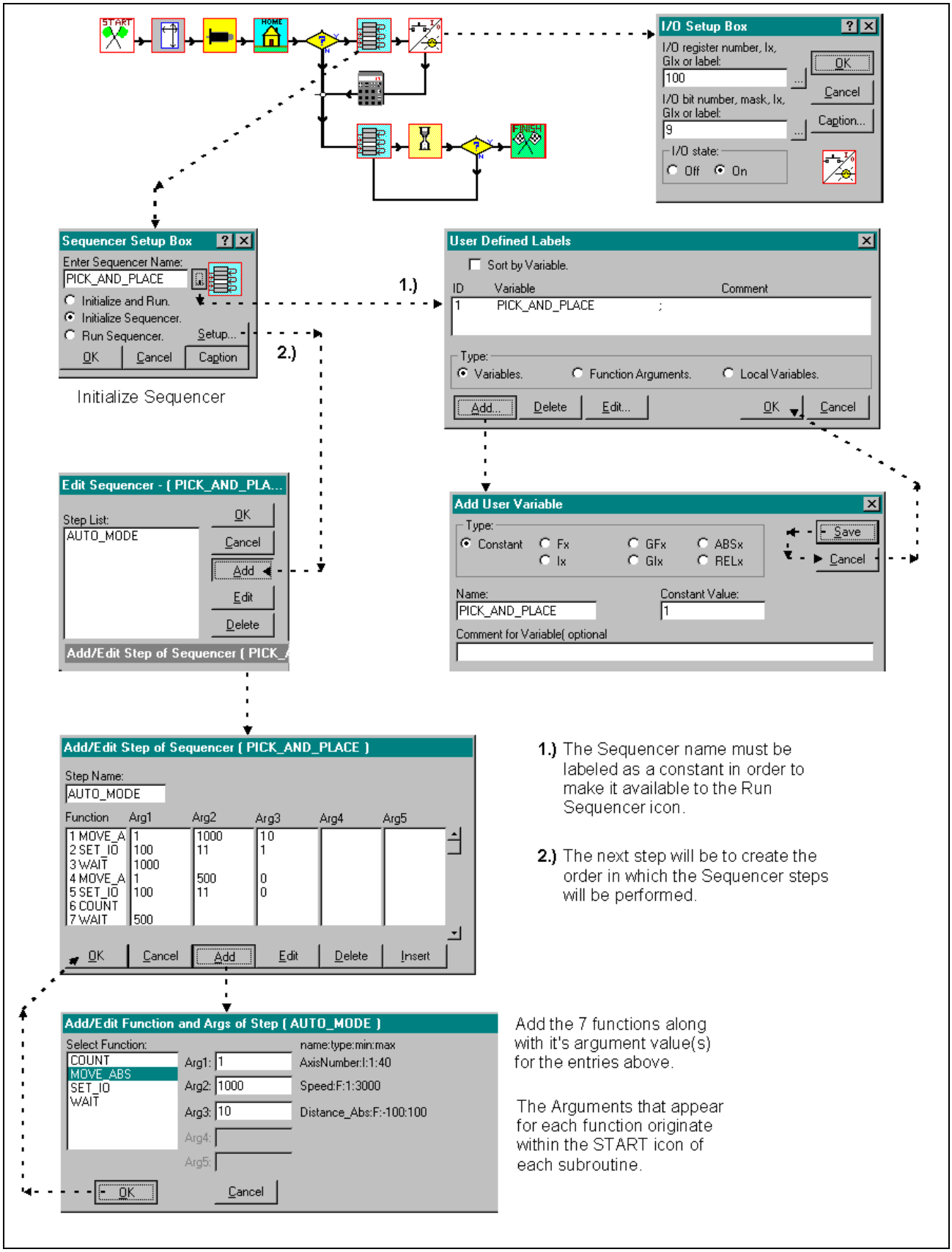


Figure D-7: Main Sequencer - Part 2 of 3

Task A, Main Sequencer - Part 3

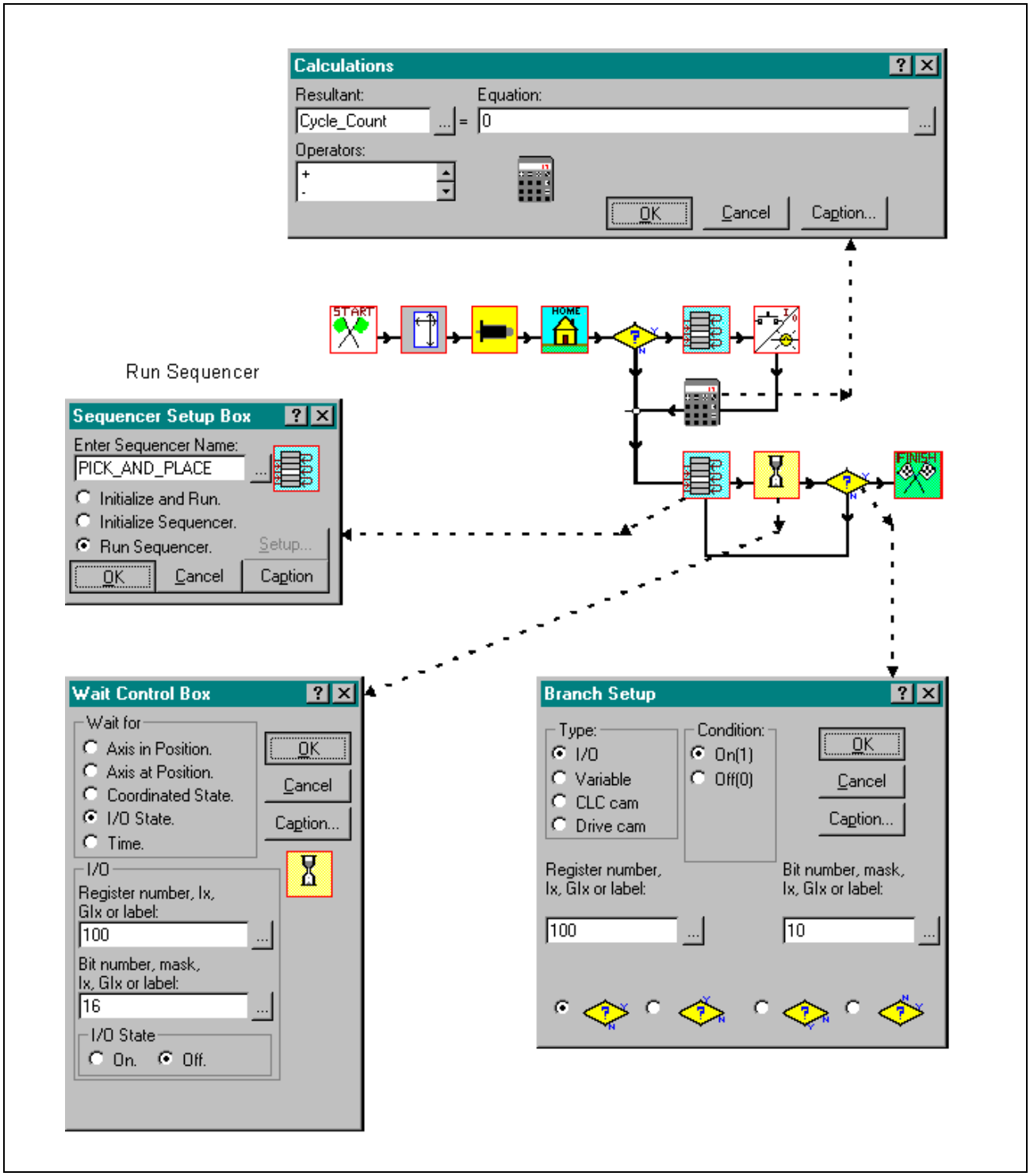


Figure D-8: Main Sequencer - Part 3 of 3

Create Bit Label

Create the following bit labels for the sequencer program. These labels will make the operation of this program much easier. Refer to ***Bit Labels on page 4-7*** for details.

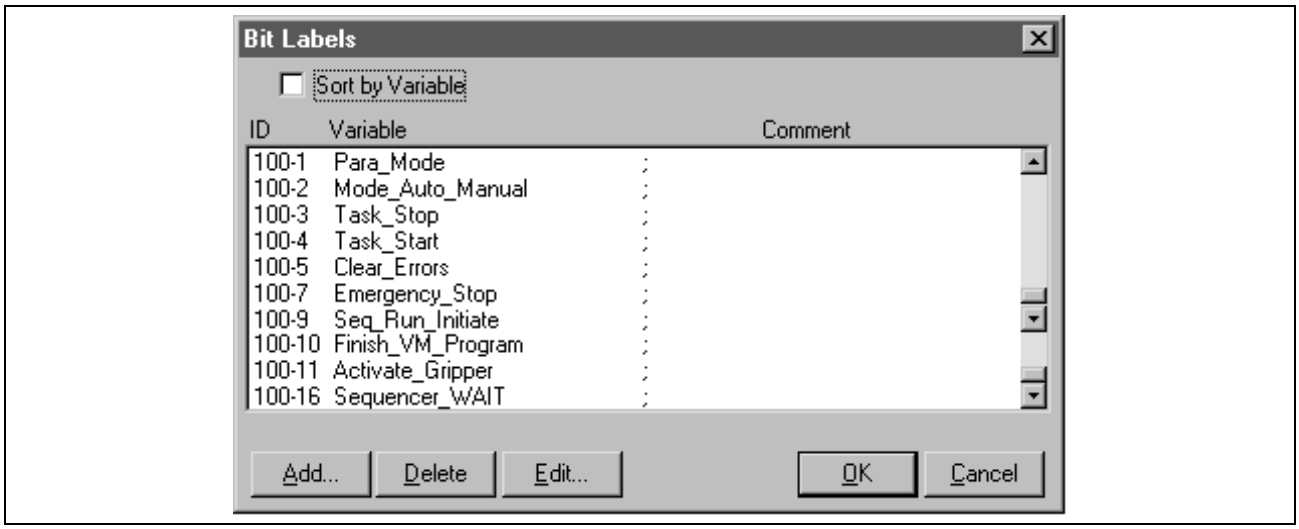
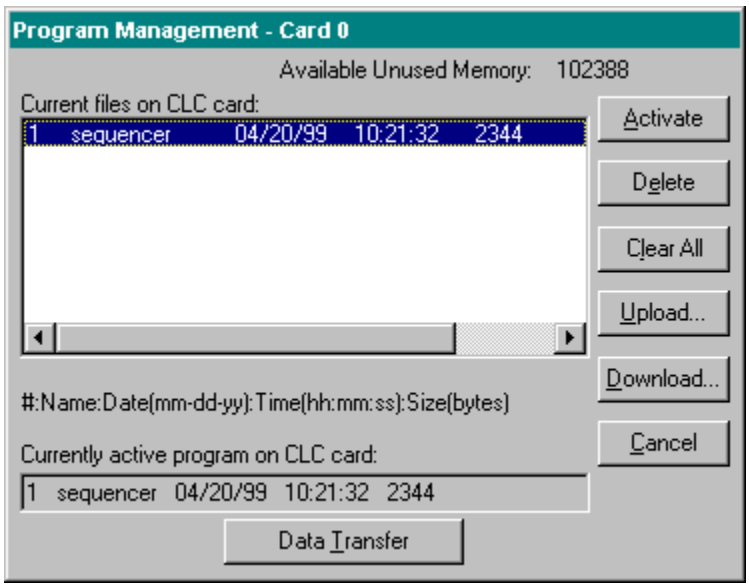


Figure D-9: Bit Labels

Save, Compile, and Download

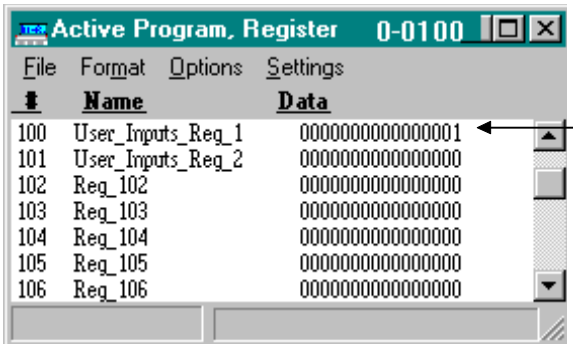
1. From VisualMotion Toolkit's main menu select ***File ⇒ Save, Compile, Download*** or click on the icon and save the program as ***Sequencer.str***

2. Once the program is compiled and downloaded, the Program Management window will open. If the sequencer program is the only program being downloaded to the CLC card, it will automatically be activated. If more than one program exists on the CLC card, highlight and activate the sequencer program by first clicking on the file name and then clicking the Activate button.



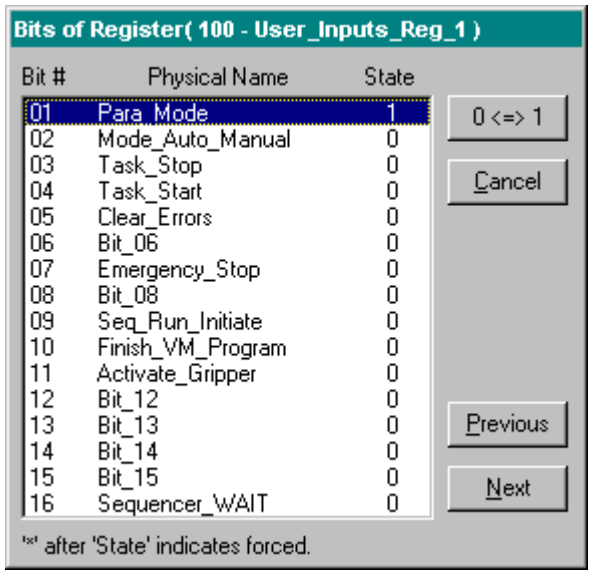
Run Sequencer Program

1. Select ***Data ⇒ Registers*** from the main menu. Scroll down to register 100 and double-click on it to open.



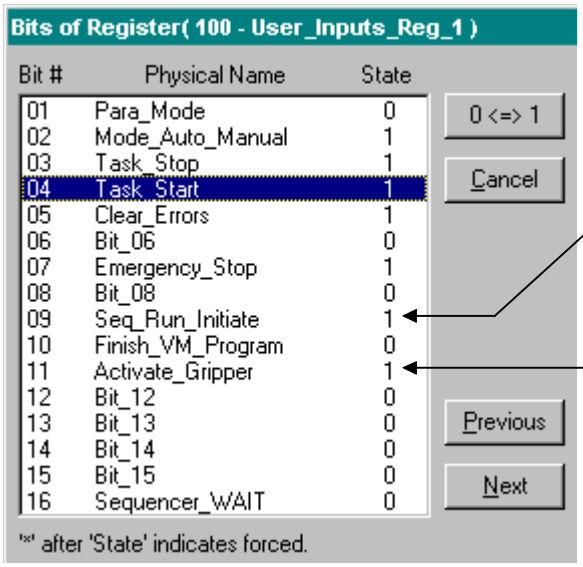
Bit (one)is high indicating that the system is in Parameter Mode.

The bits should now be listed along with their corresponding labels (Physical Name).



2. Change the state of the following bits as listed in the order shown below:

	Bit	*Label*	*State*
a.	07	Emergency-Stop	0 to 1
b.	01	Param_Mode	1 to 0
c.	05	Clear_Errors	0 to 1
d.	02	Mode_Auto_Manual	0 to 1
e.	03	Task_Stop	0 to 1
f.	04	Task_Start	0 to 1

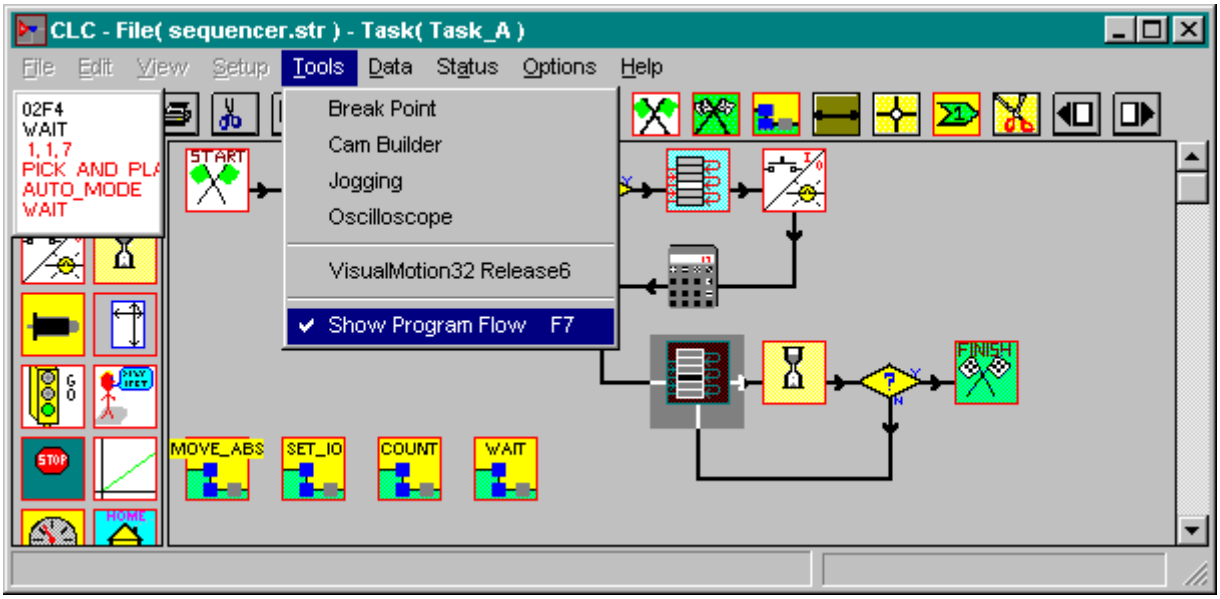


Bit 09 is normally low (0) and goes high (1) when the sequencer is first initialized.

Bit 11 is simply a visual representation of a gripper hand closing (1) and opening (0).

3. The program should now be running with the bits in this state.

To view which program icon is being processed within VisualMotion Toolkit, select ***Tools*** ⇒ ***Show Program Flow*** or press **F7** on the keyboard to turn the feature on or off.



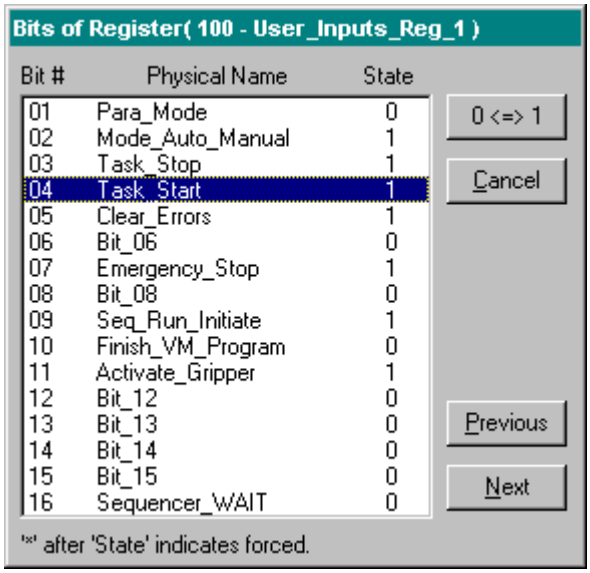
Program Operation

Changing the state (0 <=> 1) of any one of the following bits in Register 100 will stop the program from running.

Bit	*Label*	*State*
01	Param_Mode	1
02	Mode_Auto_Manual	0
03	Task_Stop	0
07	Emergency-Stop	0
09	Finish_VM_Program	1
16	Sequencer_WAIT	1

If the program was stopped using bit 100-7 *Emergency_Stop* or from another error, bit 100-5 *Clear_Errors* must be toggled from 1 to 0 to 1 before running again.

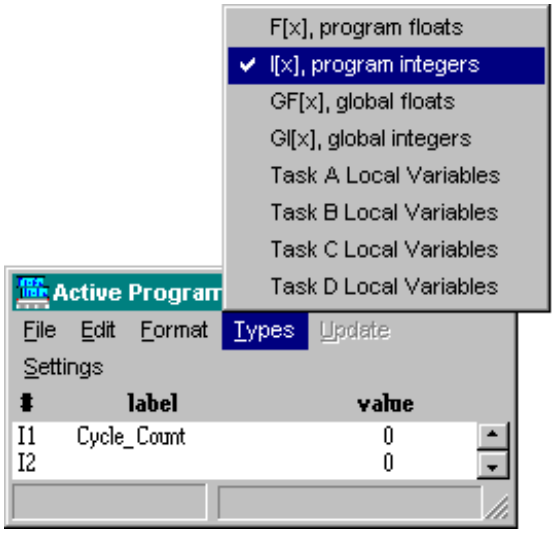
To run the program again, toggle bit 100-4 *Task Start* from 1 to 0 to 1 with the other bits in the state shown below:



Note: In order to start the program from the very beginning, toggle bit 100-2 *Mode_Auto_Manual* from 0 to 1 prior to toggling the Task Start bit. If you wish to reinitialize the sequencer, make sure that bit 09 is low (0.) Otherwise, the program will branch to the Run Sequencer icon.

The first time this program is executed, the sequencer is *initialized.* The I/O state of register 100 bit 09 goes high (1) indicating that the squencer has initialized.

The program flow will then reset a cycle counter and Run the sequencer. To view the cycle counter select ***Data ⇒ Variables*** and change the *Types* to *I[x], program integers*.

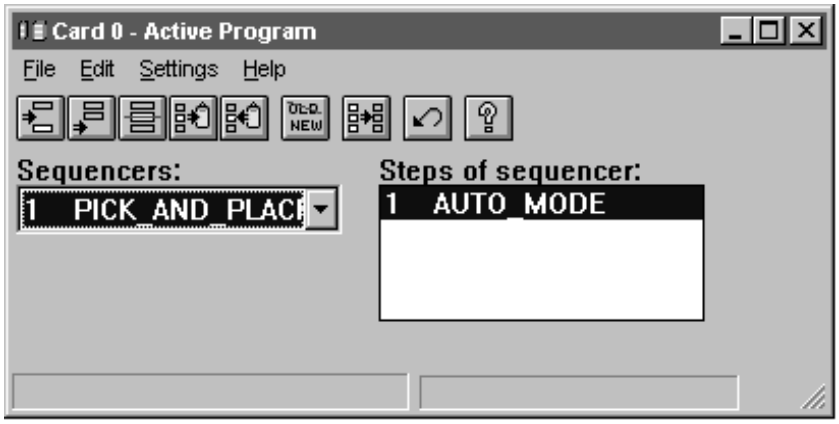


The program will continue running in the sequencer loop until a WAIT condition, register 100 bit 16, is active high(1) or finishes when bit 10 goes high (1.)

When the program is stop and restarted using bits 03 and 04, the I/O state of register 100 bit 09 will cause the program to branch to the ***Run Sequencer*** icon and continue into the sequencer loop.

Modifying Sequencer List Values

The user can make changes to the performance of the sequencer by modifying the values of each argument without having to save, compile and download the program again. With the sequencer program actively running select ***Data ⇒ Sequencer*** to open an Active Program window containing the Sequencer and Steps.



1. Double click on Step 1 - **AUTO_MODE**. The **AUTO_MODE** Step List will open as illustrated below. Any of the arguments within the function **MOVE_ABS** (absoulte move) can be changed by double clicking on one of them. Enter a different value and then select Send.

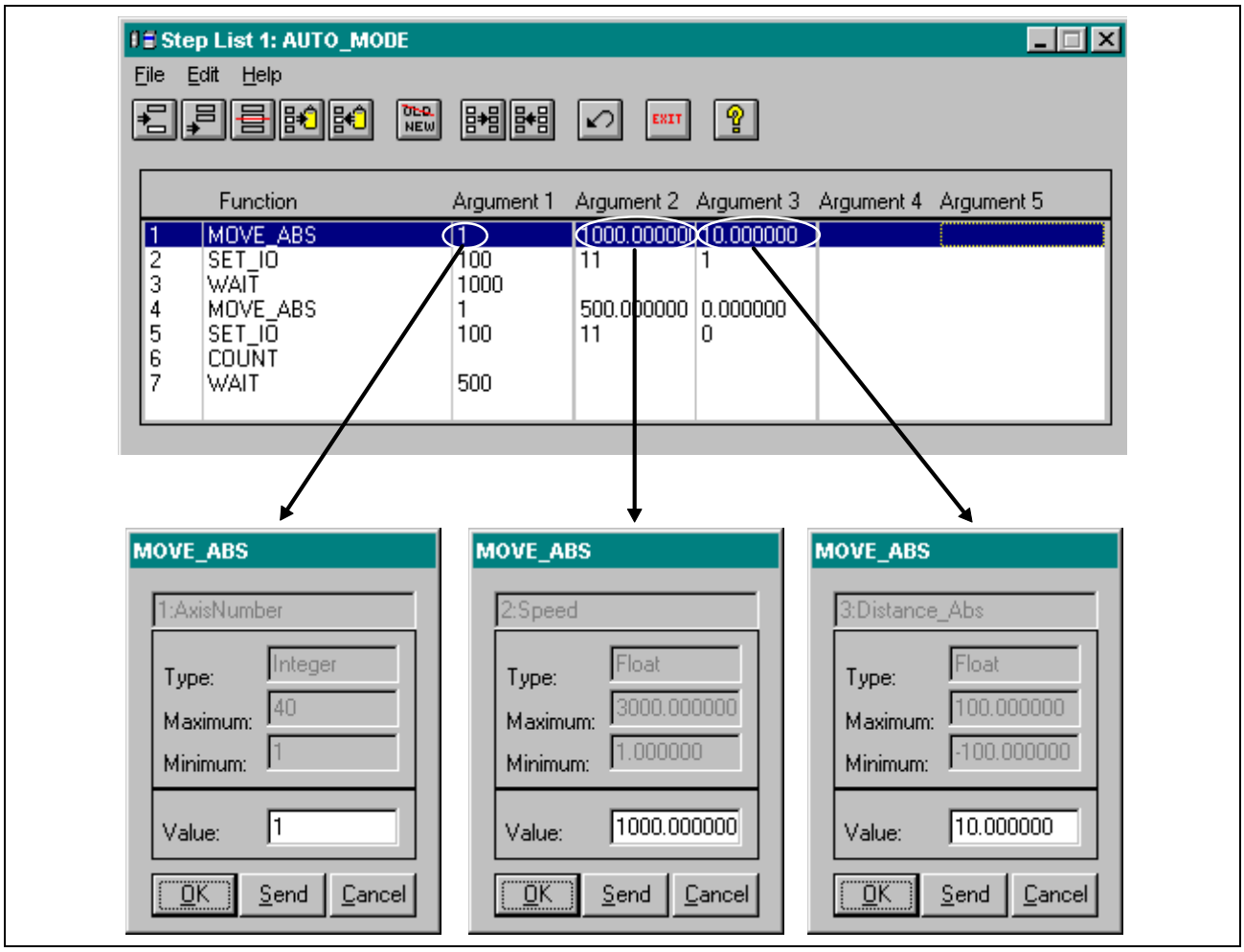


Figure D-10: Arguments within a Function

Program Icons

TASK A (Main Sequencer)

ICON	DESCRIPTION
	Standard START for all programs. No Local Variables or Function Arguments required.
	Declares the maximum number of variables, lists, steps and functions used in the Sequencer program.
	Setup for Axis includes motion type, initial acceleration and velocity and positioning mode. If "Axis Initially Halted" is selected a GO icon is required for it to run.
	The Home icon setup identifies the axis to be home by the drive controlled homing procedure.
	Setup selects "I/O" type and defines a register (100) and bit (9) to check. The condition, On(1) or Off(0), determines which branch, Y or N, the program will follow.
	The first Sequencer setup box initializes and runs "PICK_AND_PLACE". The steps initialized in this icon overwrite any Sequencer editor changes.
	The I/O setup box sets the I/O state of Register 100, bit 9 On(1)
	The “CALC” setup box resets a Cycle_Count variable to a value of zero.
	The second Sequencer setup box only runs "AUTO_MODE". The Sequencer steps must first be initialized in the Sequencer editor under the Data menu or added using the Sequencer editor.
	The “WAIT” setup box monitors the state of register (100) bit (16). The condition of bit (16), when Off(0), causes the sequencer loop to pause for a set time.
	Setup selects "I/O" type and defines a register (100) and bit (10) to check. The condition, On(1) or Off(0), determines which branch, Y or N, the program will follow.
	Standard FINISH for all programs. No optional returned values.

SUBROUTINE - MOVE_ABS

ICON	DESCRIPTION
	Standard START for all programs. Declares the name, type, minimum value and maximum value for all Function Arguments used in the subroutine.
	Velocity control box sets the axis function argument "*AxisNumber*" velocity to the function argument "*Speed"*.
	Move setup for axis function argument "*AxisNumber*." Defines the type of movement (absolute), and the distance the axis will move for function argument - "*Distance_Abs*."
	Commands the axis function argument "*AxisNumber*" to begin motion for a particular Motion Type (non-coordinated.)
	Causes the program flow to wait for the axis to be in position.
	Standard FINISH for all programs. No optional returned values.

SUBROUTINE - SET_IO

ICON	DESCRIPTION
	Standard START for all programs. Declares the name, type, minimum value and maximum value for all Function Arguments used in the subroutine.
	Branch Setup monitors the condition of the variable function argument "*State*" and compares the value before program flow continues.
	The I/O Setup Box sets the state of the function arguments "*Register*" and "*Bit*"
	Standard FINISH for all programs. No optional returned values.

SUBROUTINE - COUNT

ICON	DESCRIPTION
	Standard START for all programs. No Local Variables or Function Arguments required.
	The WAIT Control Box sets a pause in the program flow using the function argument "*Time*."
	Standard FINISH for all programs. No optional returned values.

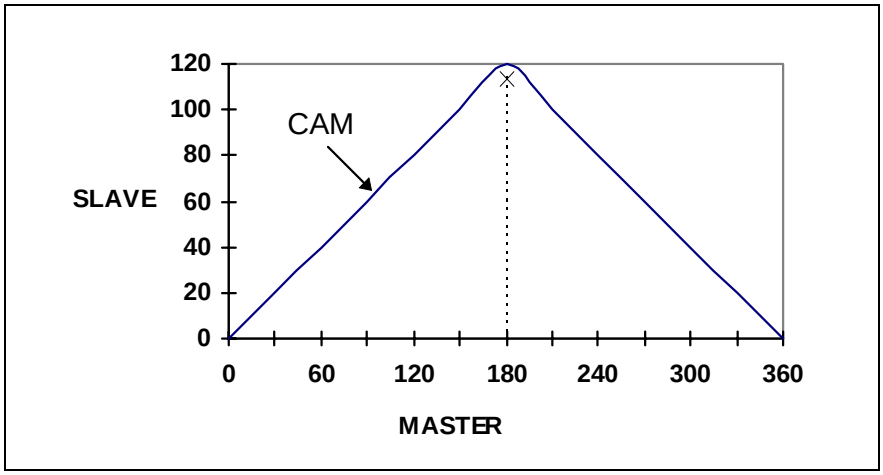
SUBROUTINE - WAIT

ICON	DESCRIPTION
	Standard START for all programs. Declares the name, type, minimum value and maximum value for all Function Argument used in the subroutine.
	The "CALC" setup box adds the equation Cycle_Count+1 to the Cycle_Count variable.
	Standard FINISH for all programs. No optional returned values.

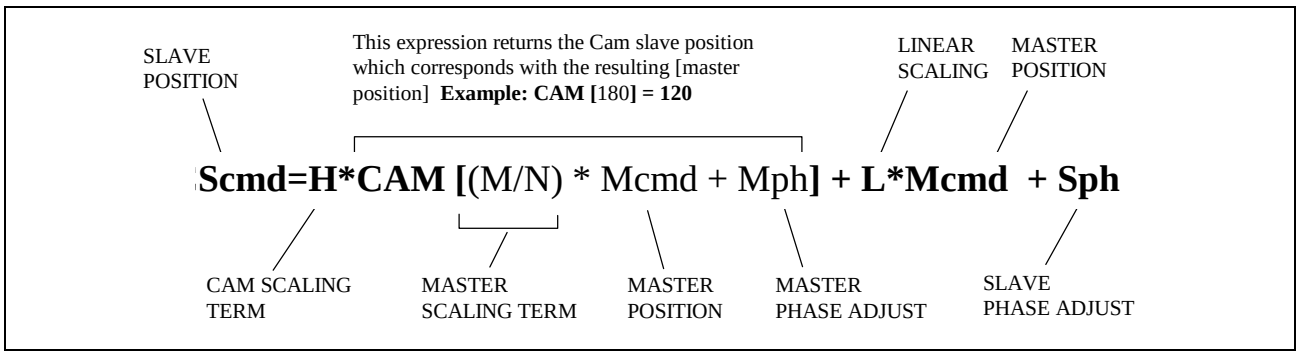
D.2 Example 2: Cam Application

The following program contains a Cam that uses an ELS virtual master and a single slave axis. The CLC uses a programmed velocity to generate the virtual master position between 0 and 360 degrees. This position is followed by the ELS slave axis.

A user defined Cam can be used to create custom motion synchronization between an ELS master and slave axis. The Cam table used in this program is a list of 1024 master and slave position points between 0 and 360 degrees. For every virtual master axis position there is a corresponding user defined slave axis position. As the virtual master position changes the ELS slave axis will move according to the corresponding position values in the Cam table. For example when the master axis position is 180º the slave axis will be at 120º. Cam [180] = 120 (refer to the equation below).



The CLC determines the proper Slave position once every SERCOS cycle using the following equation.


SLAVE POSITION
This expression returns the Cam slave position which corresponds with the resulting [master position] Example: CAM [180] = 120
LINEAR SCALING
MASTER POSITION
Scmd=H*CAM [(M/N) * Mcmd + Mph] + L*Mcmd + Sph
CAM SCALING TERM
MASTER SCALING TERM
MASTER POSITION
MASTER PHASE ADJUST
SLAVE PHASE ADJUST

If M =1, N=1, H=1 and L=0 the Slave Command Position (Scmd) will directly reflect the slave position in the cam table. If the coefficient values are changed, the resulting Scmd will be adjusted accordingly.

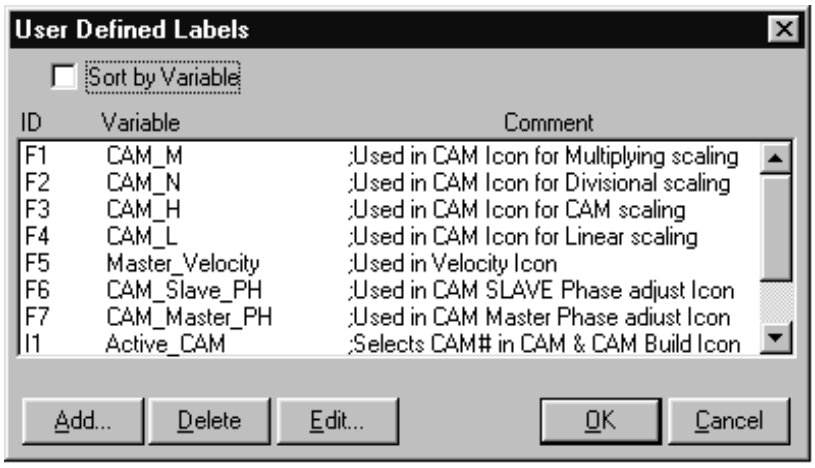
The following program will illustrate how these coefficient value adjustments will affect Cam performance. Follow the Program Instructions while referring to the Program Layout and Icon Description Sections.

Program Instructions

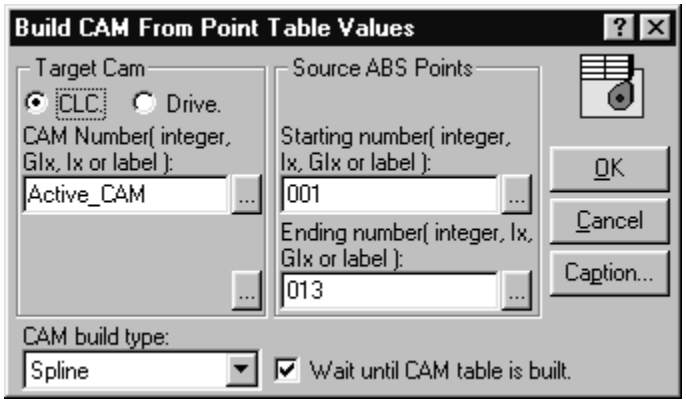
Using the Icon programming environment, create the Cam program according to the corresponding dialog boxes outlined in the Section. Before adding icons to the programming work area, save the program by selecting ***File ⇒ Save As*** and name the program CAM_App.str

1. Select **Labels ⇨ User Labels** from the Edit Menu and **Add..** the following labels:

   F1=CAM_M **F2**=CAM_N **F3**=CAM_H **F4**=CAM_L
   F5=Master_Velocity **F6**=CAM_Slave_PH **F7**=CAM_Master_PH
   I1=Active_cam


User Defined Labels
Sort by Variable
ID Variable Comment
F1 CAM_M ;Used in CAM Icon for Multiplying scaling
F2 CAM_N ;Used in CAM Icon for Divisional scaling
F3 CAM_H ;Used in CAM Icon for CAM scaling
F4 CAM_L ;Used in CAM Icon for Linear scaling
F5 Master_Velocity ;Used in Velocity Icon
F6 CAM_Slave_PH ;Used in CAM SLAVE Phase adjust Icon
F7 CAM_Master_PH ;Used in CAM Master Phase adiust Icon
I1 Active_CAM ;Selects CAM# in CAM & CAM Build Icon
Add...
Delete
Edit...
OK
Cancel

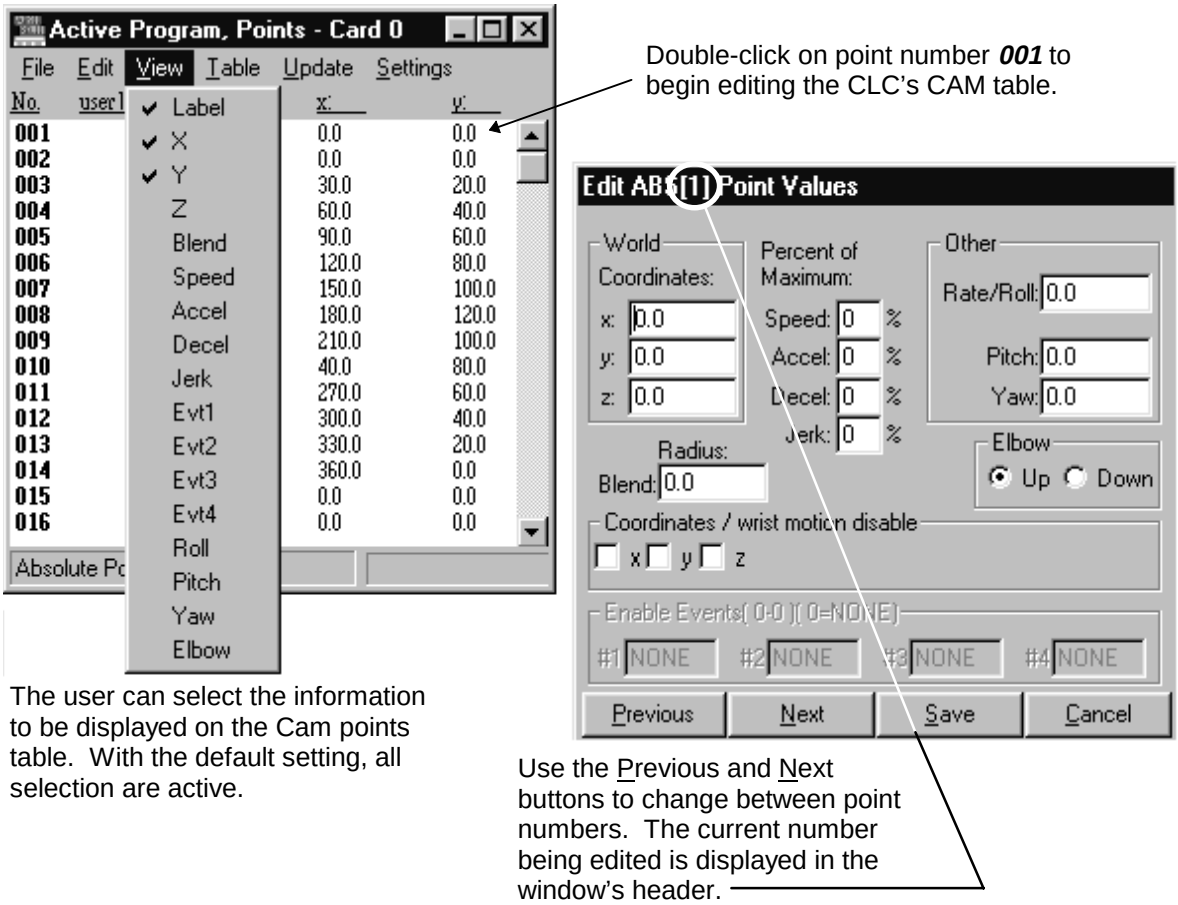
   The above window can also be displayed by clicking on the User label button . User labels can be created all at once or as required within each programming icon.



2. Save, Compile and Download the program by clicking on the button.

3. When the Program Management window appears, highlight the CAM_App program and press the Activate button. If CAM_App is the only program found on the CLC card, it will automatically be activated.

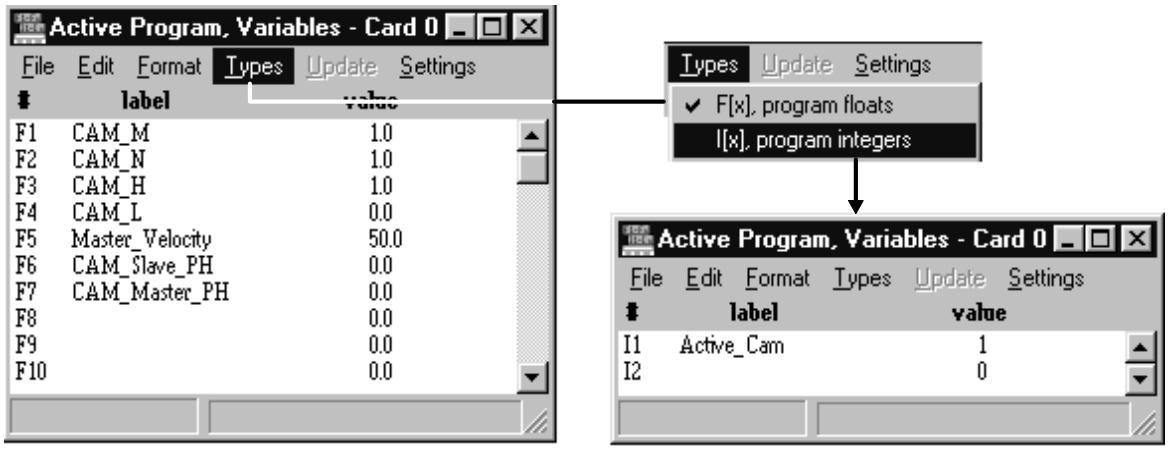
4. Create the CAM's points table by selecting ***Data ⇒ Points*** and entering the following information.



The user can select the information to be displayed on the Cam points table. With the default setting, all selection are active.

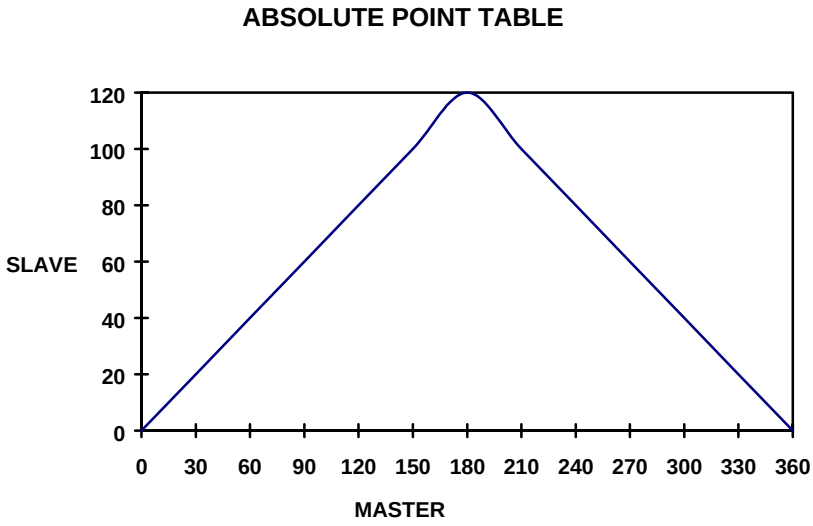
Use the Previous and Next buttons to change between point numbers. The current number being edited is displayed in the window's header.

6. Before running the program select ***Data*** ⇒ ***Variables*** and set F1-F7 and I1 to the initial values listed below. These variables can be adjusted while the program is running.



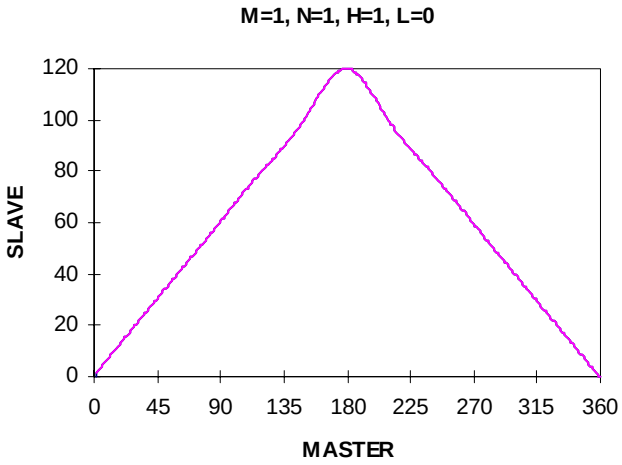
X Master	Y Slave
0	0
30	20
60	40
90	60
120	80
150	100
180	120
210	100
240	80
270	60
300	40
330	20
360	0

The absolute point table is used by the Cam Build icon to create a normalized 1024 point Cam table. The *Spline Cam build type* builds the table by connecting the points with third order splines.



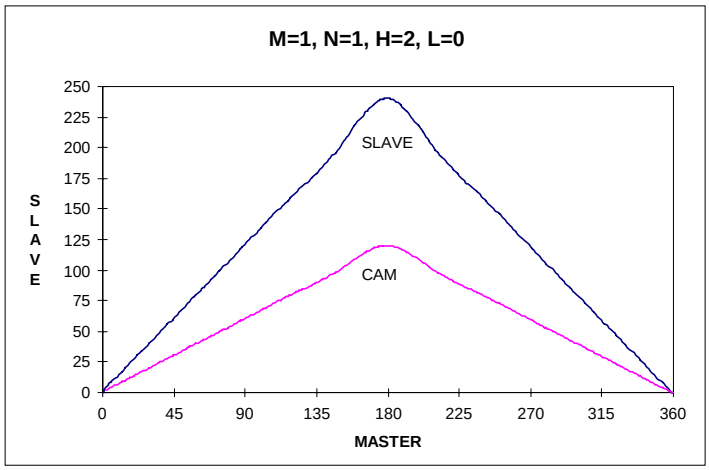
After the Cam table is built, the Cam icon uses the Cam equation to associate the Cam table to the slave axis. The coefficient variables, *defined below*, are a part of the Cam equation and will adjust the slave axis position. With the program running Adjust the variables as outlined below to see the effect they have on the slave axis.

M (CAM_M)	multiplying scaling term of the master position.
N (CAM_N)	divisional scaling term of the master position.
H (CAM_H)	cam scaling term (stretch factor).
L (CAM_L)	linear scaling term of master position.



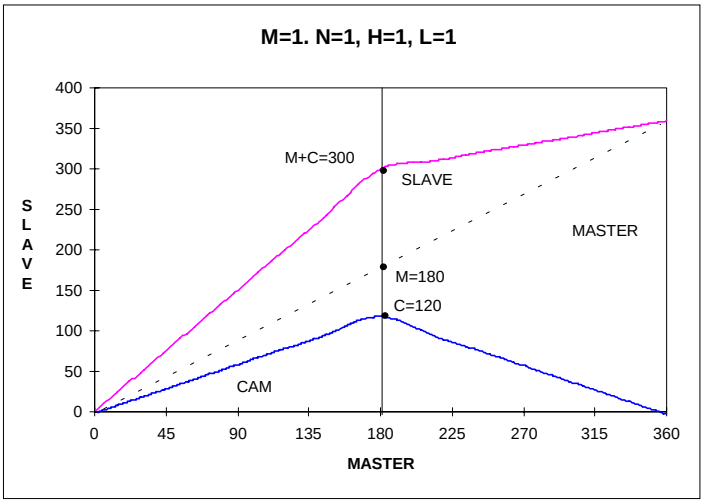
M (CAM_M)	1
N (CAM_N)	1
H (CAM_H)	1
L (CAM_L)	0

With these coefficients the slave axis will rotate back and forth between 0 and 120 degrees. The slave axis is directly following the Cam without any position adjustments.



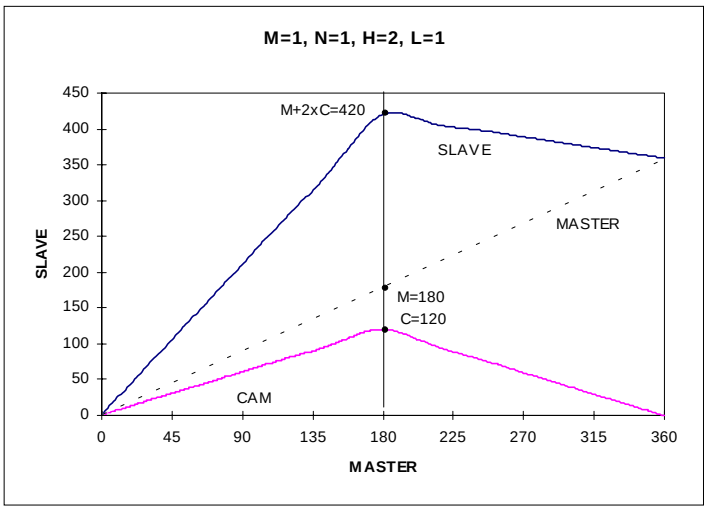
M (CAM_M)	1
N (CAM_N)	1
H (CAM_H)	**2**
L (CAM_L)	0

With these coefficients the slave axis will rotate back and forth between 0 and 240 degrees. The slave axis is following the Cam with a scale factor of two.



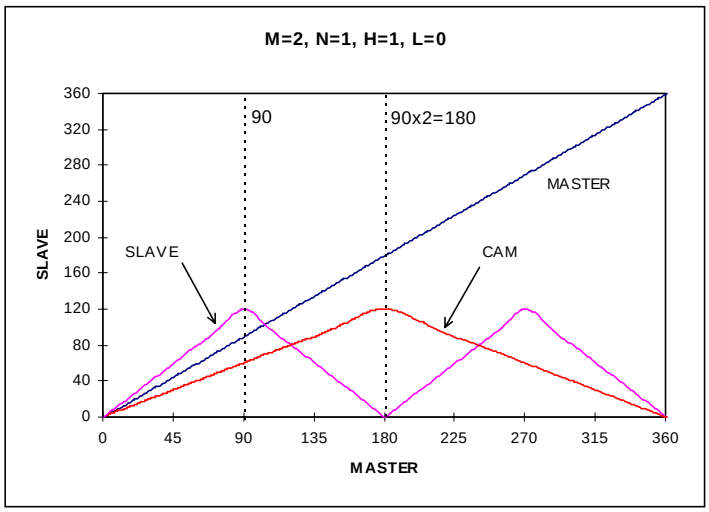
M (CAM_M)	1
N (CAM_N)	1
H (CAM_H)	1
L (CAM_L)	**1**

With these coefficients the slave axis will make a complete rotation. The slave axis will be ahead of the master by the corresponding value in the Cam table



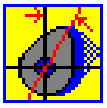
M (CAM_M)	1
N (CAM_N)	1
H (CAM_H)	**2**
L (CAM_L)	**1**

With these coefficients the slave axis will make a complete rotation. The slave axis will be ahead of the master by the corresponding value in the Cam table. The slave axis will reverse while the master is between 180 to 360 degrees. This allows the slave axis to be aligned with the master at 360 degrees.

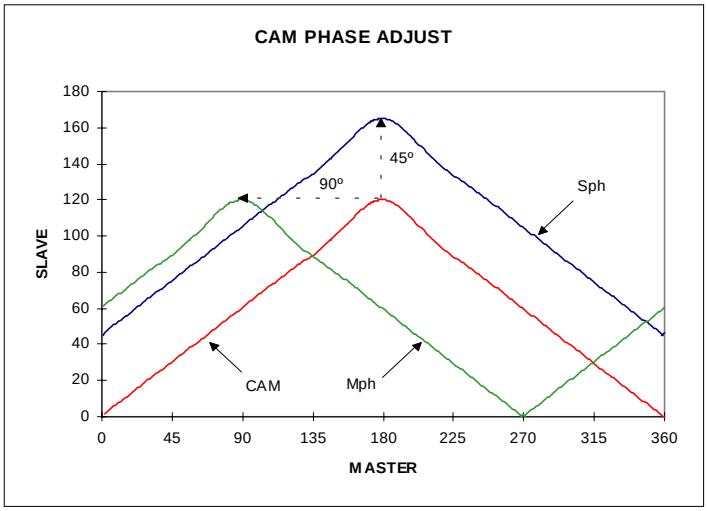


M (CAM_M)	2
N (CAM_N)	1
H (CAM_H)	1
L (CAM_L)	0

With these coefficients the slave axis will get its value from the corresponding master cam axis position scaled by a factor of two. At 90 degrees the slave axis will be in the 180 degree position. The slave axis will rotate between 0 and 120 degrees twice every Cam cycle.

Master or Slave Phase Adjust 

The Cam Phase Adjust icons are used in the program to control the Master and Slave phase adjust. The Phase adjust can be changed with the program running by changing the values of SPH (F6) and MPH (F7). Use the following values to see how the phase adjust effects the performance of the slave axis. The chart below illustrates Sph = 45º and Mph = 90º with M=1, N=1, H=1 and L=0.



Program Layout

Task A, CAM program - Part 1

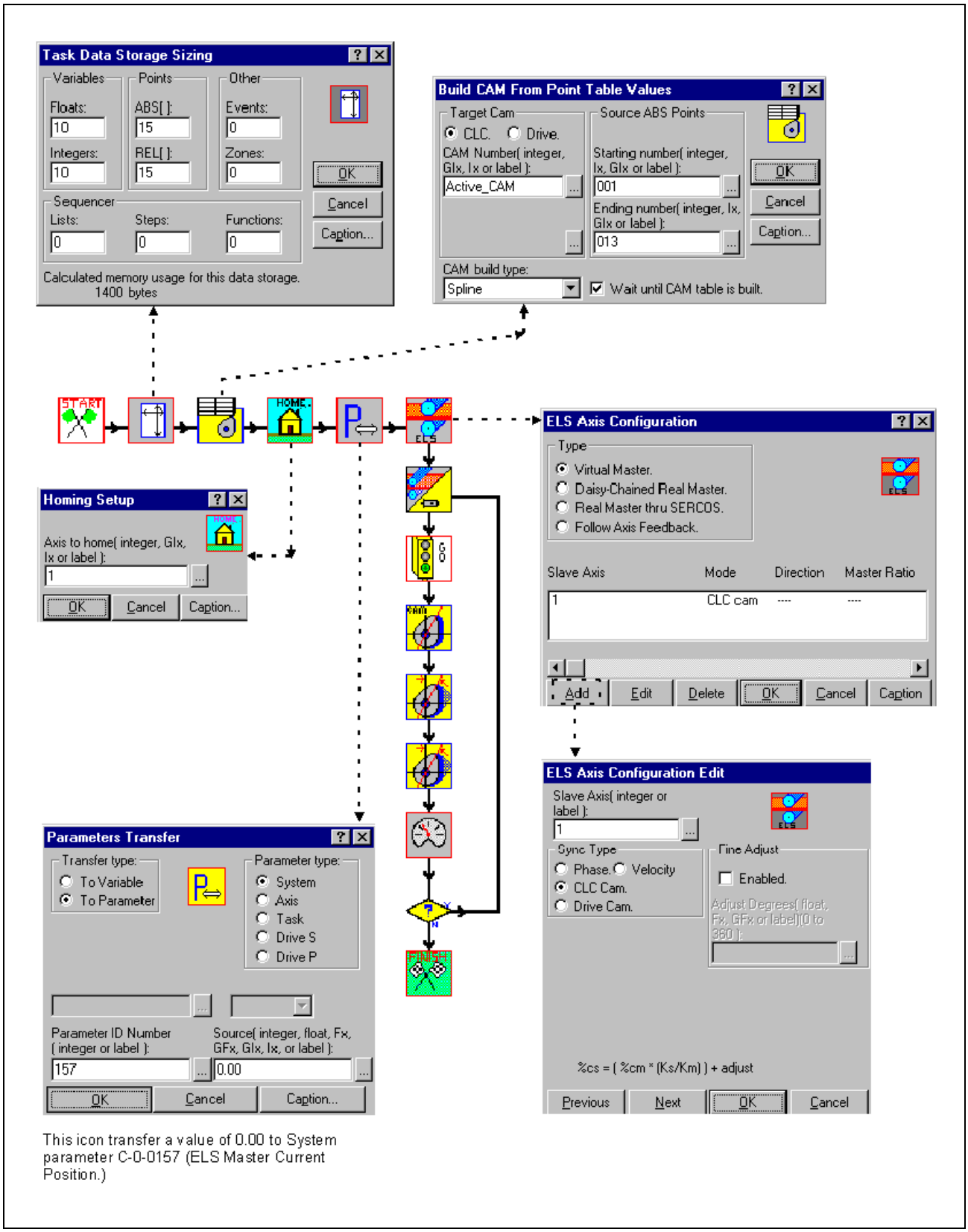


Figure D-11: CAM program - Part 1 of 2

Task A, CAM program - Part 2

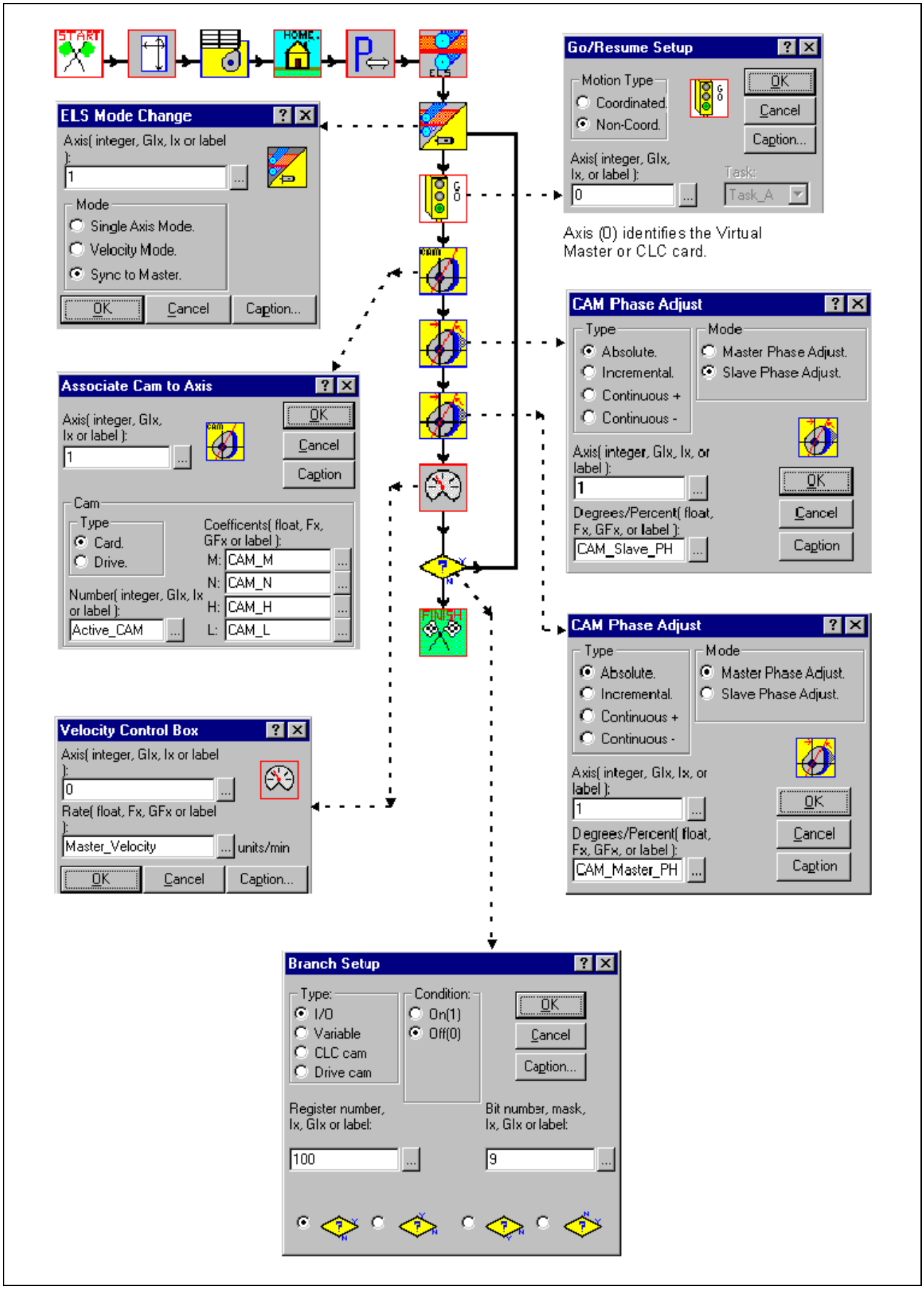


Figure D-12: CAM program - Part 2 of 2

CAM Program Icons

TASK A

ICON	DESCRIPTION
	Standard START for all programs. No local variables required.
	Declares the maximum number of points allocated for point table. Point table used to build CAM. Sample program declares: ABS [9], REL [9].
	Used to Build a CAM from a point table. Assign CAM number, CAM type and range of source points used to build CAM. Select wait until CAM is built.
	Moves the slave axis (1) to it's home position.
	Sets the System Parameter C-0-0157 to 0.00. This resets the Virtual Master to it's home position.
	Selects the type of Master/Slave Axis used. Sample program uses Virtual Master (Axis 0). Add/Edit Slave Axis: number (1), sync type (cam), fine adjustments (if required).
	Defines the ELS mode which causes a slave axis (1) to sync. to the master axis according to the cam table as defined in the ELS axis configuration ICON.
	GO starts the virtual master according to the acceleration, velocity and deceleration values defined in their respective icons.
	Associates Cam (1) to Axis (1). Defines coefficients used in cam equation.
	Selects which Cam phase adjust to perform and starts the phase adjust.
	Defines the velocity for the Virtual Master (Axis 0).
	Branch setup which checks an unused I/O register to create a loop in program.
	Standard FINISH for all programs. No optional returned values.

D.3 Example 3: Programmable Limit Switch (PLS) Application

The following sample program contains a PLS (Programmable Limit Switch) which includes an ELS master and slave axis. As the ELS virtual master position changes the PLS will turn the output register 129, bits 1-16 on and off. The ELS slave axis will rotate to reflect the position of the ELS master.

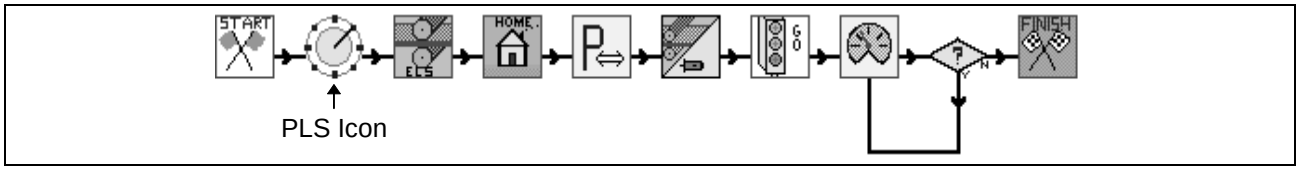


Figure D-13: programmable limit switch program

Program Instructions

Using VisualMotion Toolkit, create a new program by selecting ***File*** ⇒ ***New*** and select *Icon* as your programming work area.

1. Before adding icons to the programming work area, save the program by selecting ***File*** ⇒ ***Save As*** and name the program PLS_App.str

2. Create the variable *F1 - Master_Vel* by selecting ***Edit*** ⇒ ***Labels*** ⇒ ***User Labels***. This variable will be used to control the velocity of the master.

3. Using the Program Layout on page D-32, create the PLS program using the information contained within each icon's setup box.

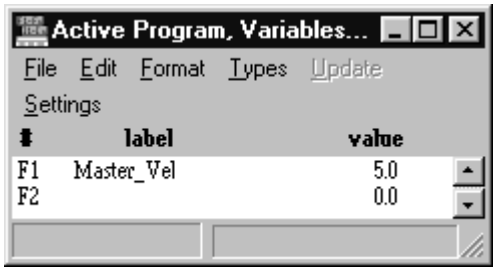
4. Save, Compile and Download the program by clicking on the button.

5. When the Program Management window appears, highlight the PLS_App program and press the Activate button. If PLS_App is the only program found on the CLC card, it will automatically be activated.

Note: ELS applications running PLS (Programmable Limit Switches) require that the *Minimum SERCOS Cycle Time,* Card Parameter C-0-0099 be increased from 2000 msec to 4000 msec.

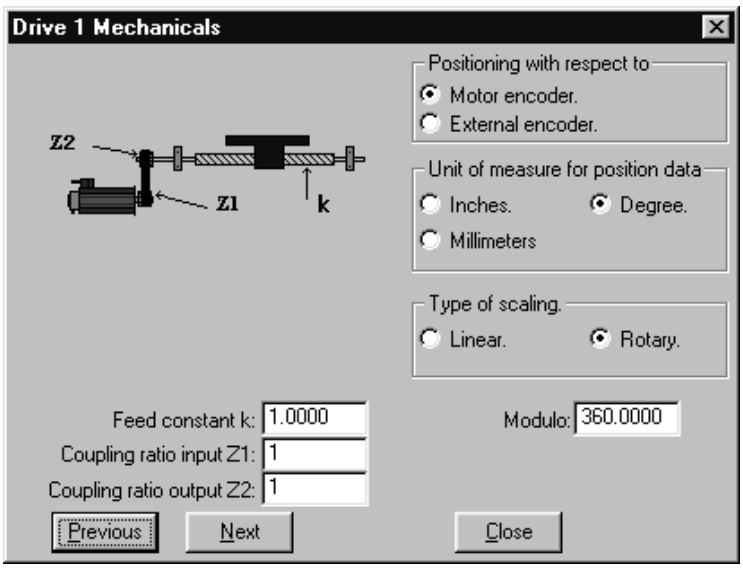
---

6. Before the PLS program can be run, select ***Data*** ⇒ ***Variables*** and enter a value for the F1 variable Master_Vel. Enter a value of 5.



Note: The program will not run unless a value exist for the Master_Vel variable. The motor is simply a slave and must respond to a velocity given to by the master.

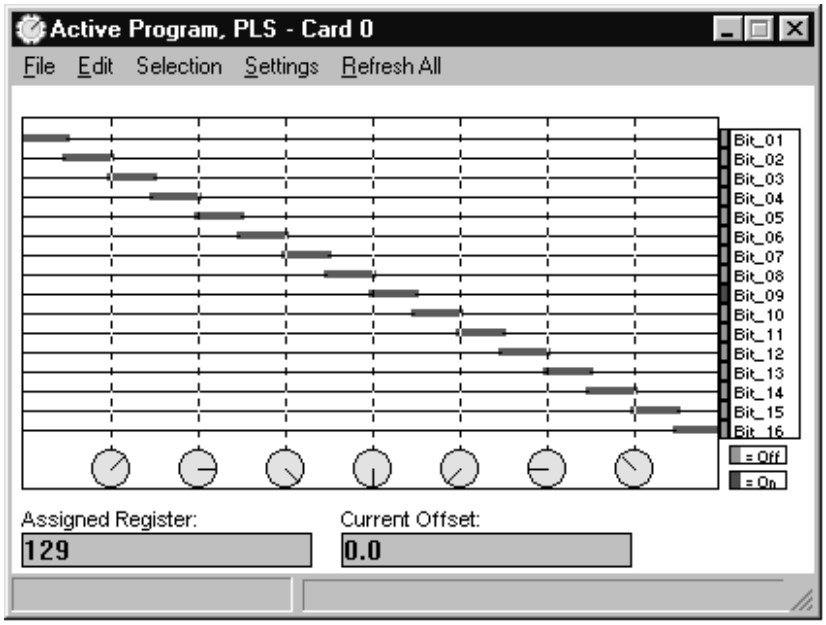
Note: Verify that the mechanical settings of the slave axis are as follows:
Select ***Status ⇒ Drives*** and the ***Parameters ⇒ Mechanical***



7. Using the default I/O Mapper file *Def100.iom* found in the folder structure ***Indramat - clc - Param***, run the program by setting register 100's bits as follows.

	Bit	***Label***	***State***
a.	07	Emergency-Stop	0 to 1
b.	01	Param_Mode	1 to 0
c.	05	Clear_Errors	0 to 1
d.	02	Mode_Auto_Manual	0 to 1
e.	03	Task_Stop	0 to 1
f.	04	Task_Start	0 to 1

8. Open the PLS window by selecting ***Data ⇒ PLS***. The active program PLS window should show the bit status of the assigned PLS register as the virtual master changes position.



Program Layout

Task A, Programmable Limit Switch program

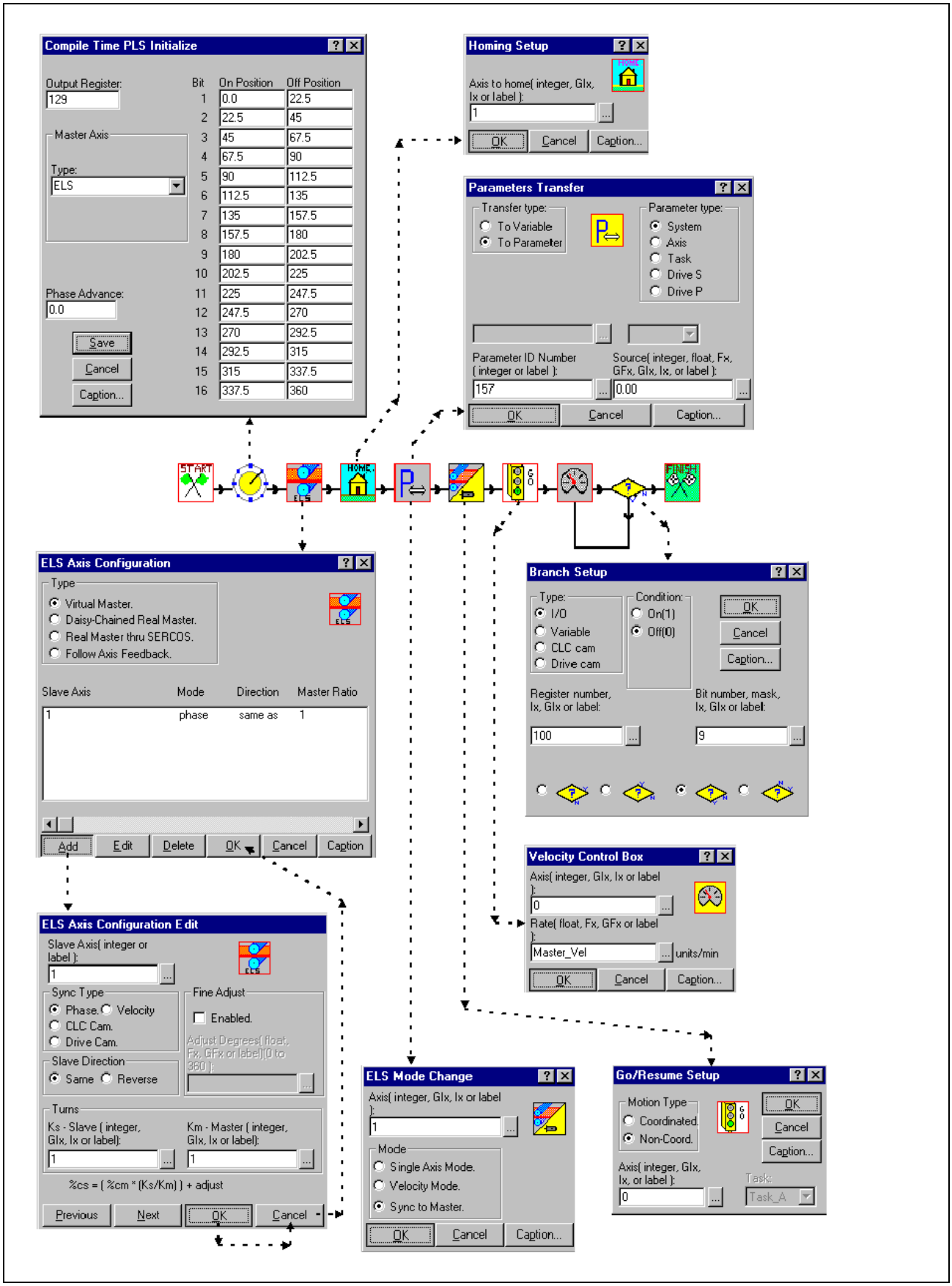


Figure D-14: Programmable Limit Switch program

PLS Program Icons

TASK A

	Standard START for all programs. No local variables required.
	Initializes the PLS. Assigns the position switch output to an unused register. Selects the type of master axis to follow. Sets up to 16 on and off switch positions with the option of phase advance.
	Selects the type of Master/Slave Axis used. Sample program uses Virtual Master (Axis 0). Add/Edit Slave Axis: number (1), sync type (cam), fine adjustments (if required).
	Moves the slave axis (1) to its home position.
	Sets the System Parameter C-0-0157 to 0.00. This resets the Virtual Master to its home position.
	Defines the ELS mode which causes a slave axis (1) to sync. to the master axis according to the cam table as defined in the ELS axis configuration ICON.
	GO starts the virtual master according to the acceleration, velocity and deceleration values defined in their respective icons.
	Defines the velocity for the Virtual Master (Axis 0). Sample program sets it at F1 units/min.
	Branch setup which checks an unused I/O register to create a loop in program.
	Standard FINISH for all programs. No optional returned values.

D.4 ECODRIVE 3 PLS Function

Indramat's ECODRIVE 03 series digital controllers include a 16 channel PLS (also referred to as a CAM Switch.) The individual ON, OFF and Lead-times can be adjusted using the PLS tool in VisualMotion. The ECODRIVE 03 also has the added functionally of allowing two digital outputs to be configured to output the state of up to two PLS channels.

Note: The DKC02.3 (SERCOS) configurable digital outputs are located on X3, pins 8 and 10 with pin 9 as 0V reference.

This section provides information on how to implement the drive's PLS functionality using the two digital outputs on X3 for ESM2.1-SGP-01VRS firmware.

Configuring Signal Status Word

To remap the PLS to the drive's digital outputs, the user must adjust several factory default parameter settings that are configured to output a Warning and Error state. SERCOS parameters S-0-0026 and S-0-0328 are used to configure the Signal Status Word S-0-0144.

S-0-0026 ***Configuration List Signal Status Word*** (List of Parameter ID numbers)

This parameter contains a list of ID numbers (i.e., P-0-0135) that are mapped to the signal status word (S-0-0144). The order of the ID numbers in this configuration list determines the bit numbering, beginning with the LSB in the *Assign List Signal Status Word*.

S-0-0328 ***Assign List Signal Status Word*** (List of Bit Numbers for S-0-0026)

This parameter contains a list of bit numbers that correlates to each ID number listed in S-0-0026. Each bit number assigned in S-0-0328 must have a corresponding ID number in S-0-0026.

S-0-0144 ***Signal Status Word***

This parameter contains the binary output status of the parameter and bit numbers listed in S-0-0026 and S-0-0328. It's a 16 bit word where bit 0 of S-0-0144 is mapped to X3.8 and bit 1 is mapped to X3.10 with pin 9 as 0V reference on the DKC02.3.

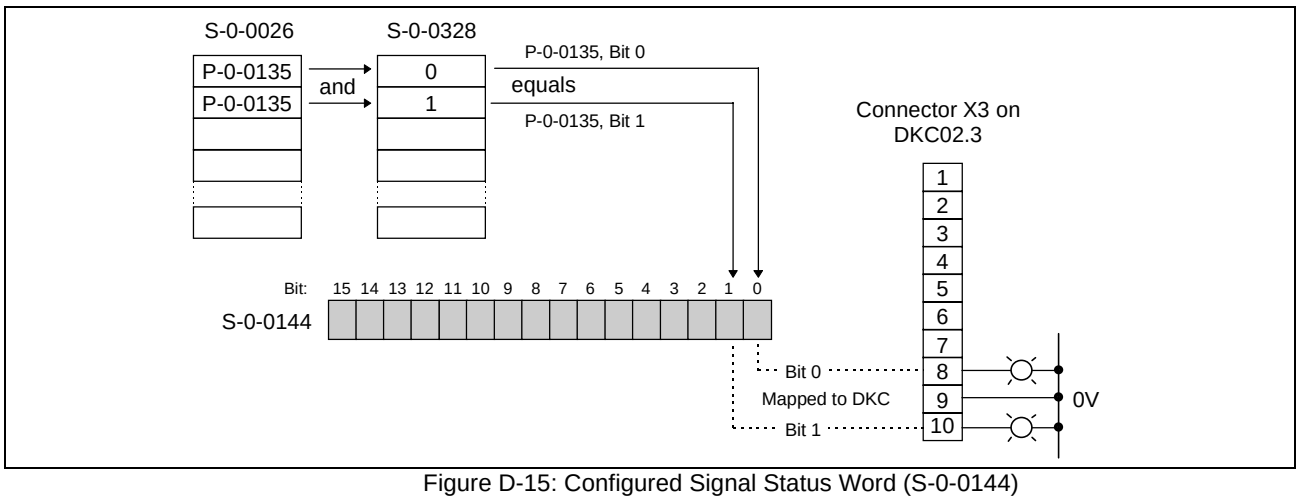
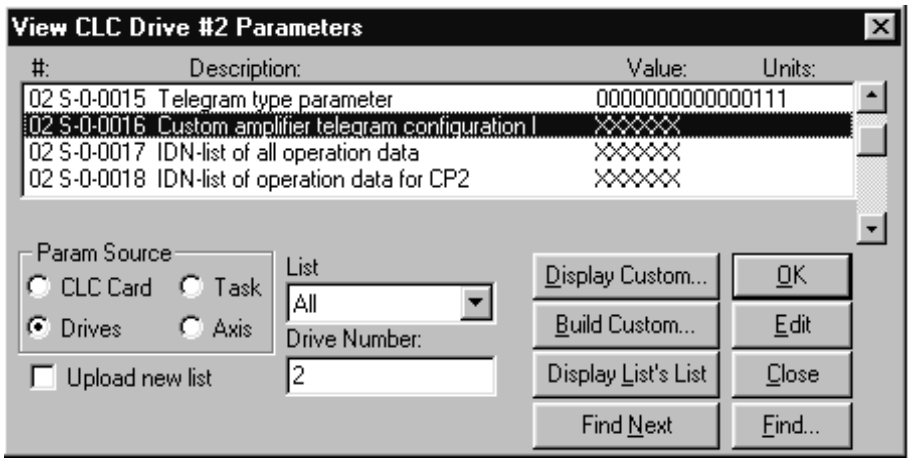


Figure D-15: Configured Signal Status Word (S-0-0144)

SERCOS Parameter

VisualMotion's parameter overview function can be used to view the drive's SERCOS parameters; however, VisualMotion currently cannot modify a SERCOS parameter containing a list of parameters.

For example; a SERCOS parameter containing a list of parameters appears as six X's.

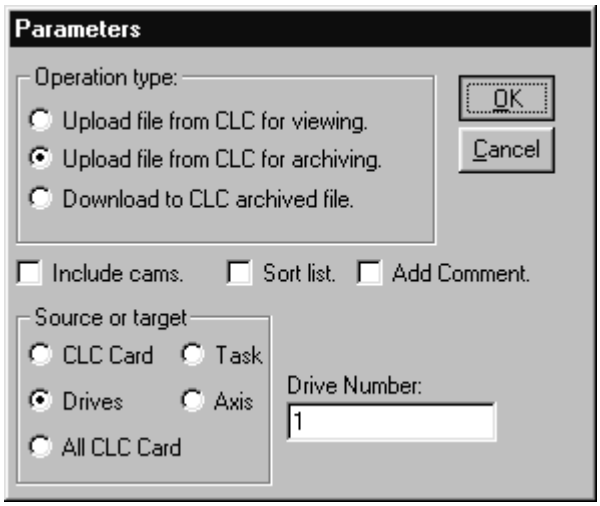


The modification of a SERCOS list parameter must be done manually using a text editor such as notepad and then downloaded to the desired digital drive using VisualMotion Toolkit.

Modifying a Parameter List

Since the modification of parameters S-0-0026 and S-0-0328 is not possible using VisualMotion Toolkit, follow the steps to ensure proper backup and modification of these parameters.

1. Create an archive of the drive that will be modified. Start VisualMotion Toolkit and select ***File ⇒ Transfer Parameters***. Select "*Upload file from CLC for archiving"* and the desired drive and press *OK* and name the file.



2. Open a text editor such as Notepad and create a file using the following format. Refer to the VisualMotion 6.0 Reference Manual, Appendix A for detailed information on the ACSI protocol described below.

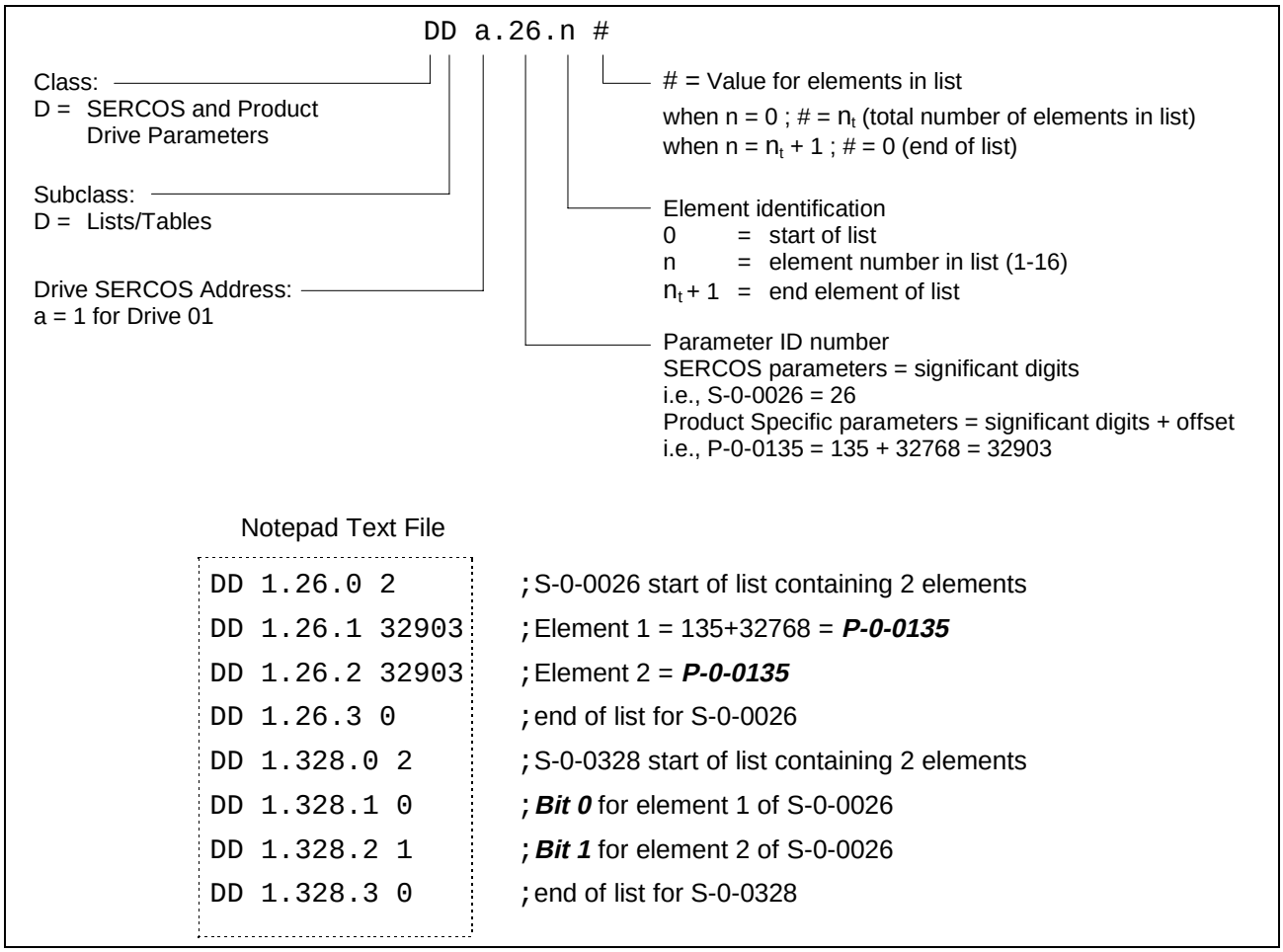
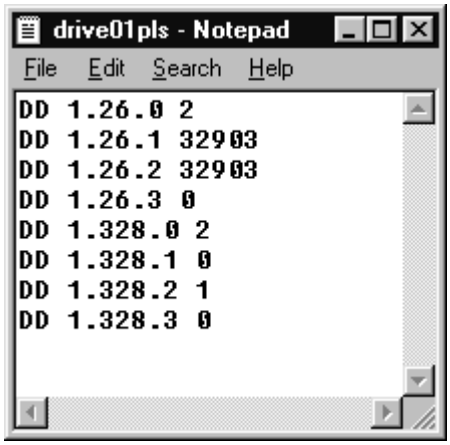

DD a.26.n #
Class:
D = SERCOS and Product Drive Parameters
Subclass:
D = Lists/Tables
Drive SERCOS Address:
a = 1 for Drive 01
# = Value for elements in list
when n = 0 ; # = n_t (total number of elements in list)
when n = n_t + 1 ; # = 0 (end of list)
Element identification
0 = start of list
n = element number in list (1-16)
n_t + 1 = end element of list
Parameter ID number
SERCOS parameters = significant digits
i.e., S-0-0026 = 26
Product Specific parameters = significant digits + offset
i.e., P-0-0135 = 135 + 32768 = 32903
Notepad Text File
DD 1.26.0 2 ;S-0-0026 start of list containing 2 elements
DD 1.26.1 32903 ;Element 1 = 135+32768 = P-0-0135
DD 1.26.2 32903 ;Element 2 = P-0-0135
DD 1.26.3 0 ;end of list for S-0-0026
DD 1.328.0 2 ;S-0-0328 start of list containing 2 elements
DD 1.328.1 0 ;Bit 0 for element 1 of S-0-0026
DD 1.328.2 1 ;Bit 1 for element 2 of S-0-0026
DD 1.328.3 0 ;end of list for S-0-0328

Figure D-16: Signal Status Word Configuration

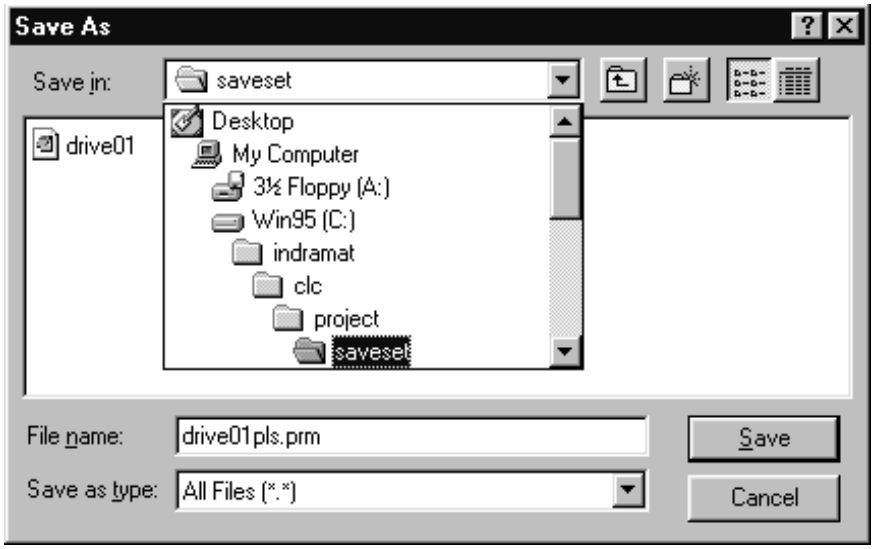


The above Notepad text file configures the Signal Status Word (S-0-0144) as follows:

The state of parameter P-0-0135, bit 0 is directly mapped to connector X3 pin 8

The state of parameter P-0-0135, bit 1 is directly mapped to connector X3 pin 10

3. Save the text file as *Drive01pls.prm* under Indramat\clc\saveset.



4. Start VisualMotion if closed and open the desired program to use with the PLS function of the DKC02.3.
5. Select ***File ⇒ Transfer Parameters.*** Select "*Download to CLC archived file",* the desired drive and press *OK*. Select the "*Drive01pls.prm"* file and press *Open*

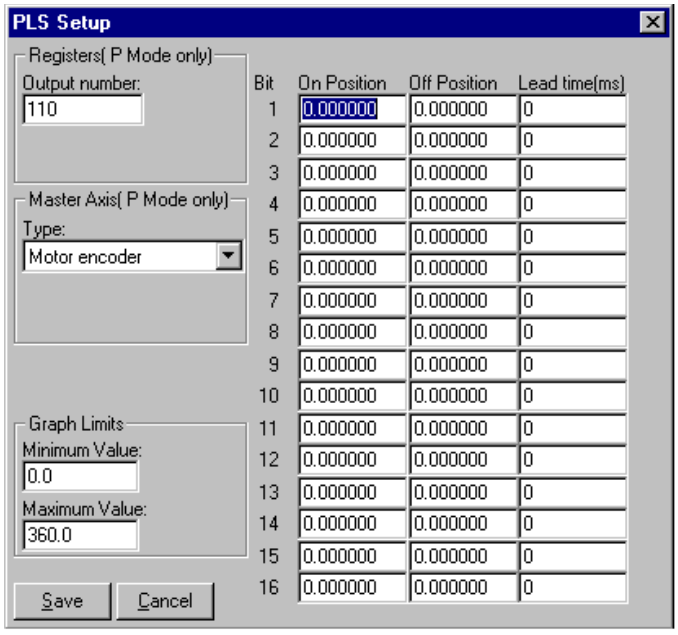
Programming PLS ON/OFF Signal

VisualMotion Toolkit will be used to select a Master Axis as well as the ON and OFF positions, including lead times, for the PLS function using bits 1 and 2.

6. Select ***Data ⇒ PLS,*** then ***Selection ⇒ Drive 1*** and enter the ON and OFF positions for Bits 1 and 2.

Master Axis type is directly mapped to P-0-0131.

P-0-0131 options:
Disable = 0
Motor Encoder = 1
External Encoder = 2



On Position bits 1-16 are mapped directly to P-0-0132, bits 0-15.

Off Position bits 1-16 are mapped directly to P-0-0133, bits 0-15

Lead time bits 1-16 are directly mapped to P-0-0134, bits 0-15

Bits 1 and 2 on this screen are mapped to P-0-0135, bits 0 and 1 respectively.

7. After entering the ON, OFF, and Lead times for bits 1 and 2, press *Save* and Run Program.

D.5 Demonstration Program

Linear Motion Application Demos

The following two demos consists of a Single Axis and a Coordinated Motion Application. Both application demos automatically setup all the required variables and values. The user simply starts the programs using the appropriate user registers and bits.

Setup

In order to run this demonstration package you need to have the following files:

Demo_1.str	Icon program file displayed in VisualMotion
Demo_2.str	Icon program file displayed in VisualMotion

To open and activate each demo, use the following steps.

Note: The two demo files are installed with VisualMotion Toolkit version V12. For customers with earlier versions of VisualMotion Toolkit, copies of the demo files can be requested by contacting a CLC Application engineer.

1. Once the files have been obtained, copy both demo programs to the following directory:
   Indramat\clc\project

2. Start the VisualMotion Toolkit program. From the main menu select ***File ⇒ Open*** and select Demo_1.str.

3. The next step will be to Save, Compile and Download “Demo_1” to the CLC card. Click on the button. Once the file has been downloaded to the card, select it and then click on the “Activate” button.

4. Use the default I/O Mapper (Def100.iom) found in directory ***Indramat/clc/param.*** Refer to *I/O Mapper* on page 4-3.

5. The run each demo program, select register 100 from ***Data ⇒ Registers*** and change the state of the following bits.

	Bit	Label	State
a.	07	Emergency-Stop	0 to 1
b.	01	Param_Mode	1 to 0
c.	05	Clear_Errors	0 to 1
d.	02	Mode_Auto_Manual	0 to 1
e.	03	Task_Stop	0 to 1
f.	04	Task_Start	0 to 1
g.	10	Motion start Bit	0 to 1

Refer to Run Sample Program on page 4-9 for examples.

VisualMotion Icon Language Program - Demo_1 (Single Axis)

Demo_1 is a Single Axis program that when started using register 100, loads and sets up the required variables and axis and waits for the activation of register 100 bit 10 to begin motion.

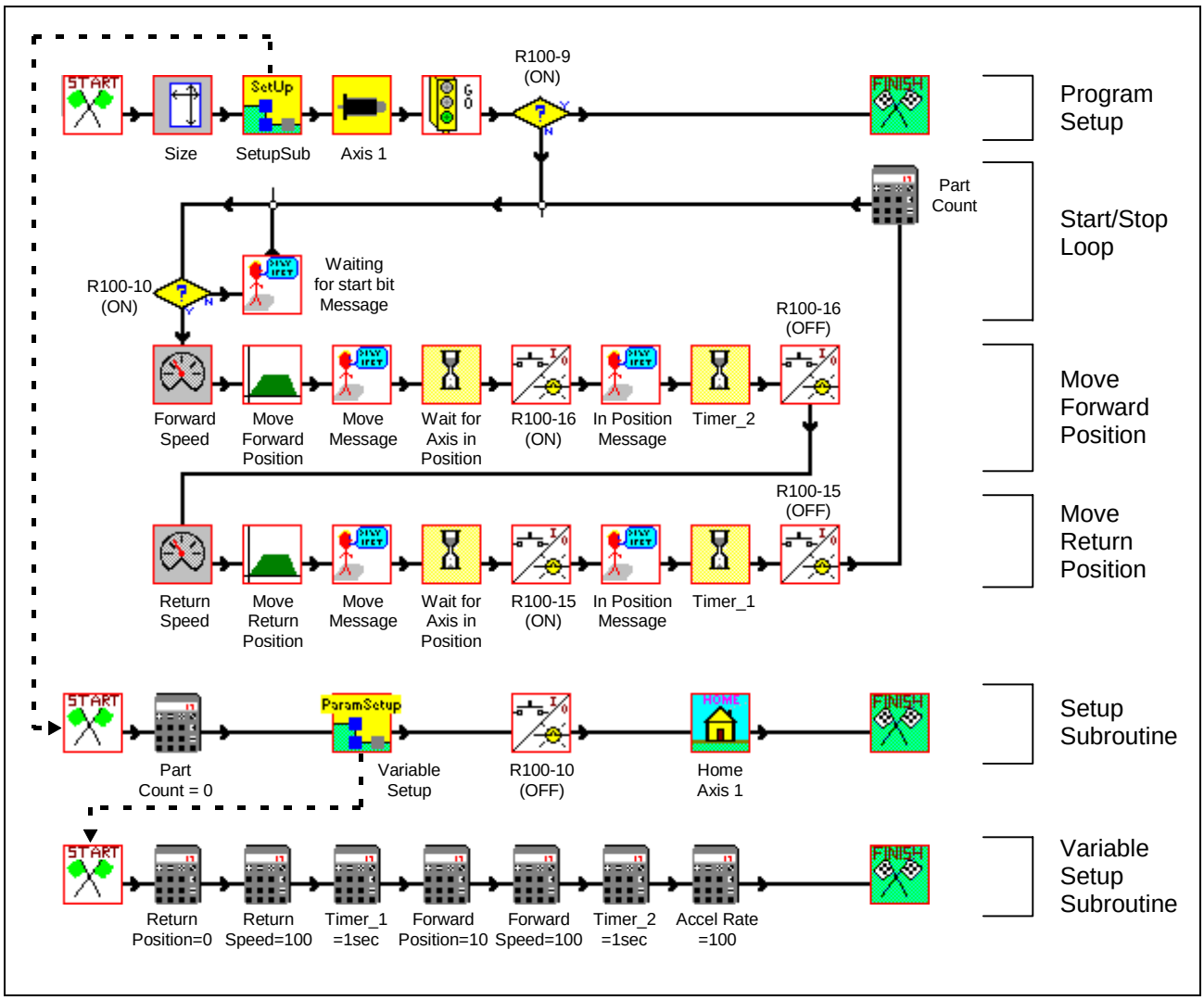
Once motion begins, values for the variables can be modified by selecting ***Data ⇒ Variables*** and changing the Program Variables.

Single Axis Program Variables

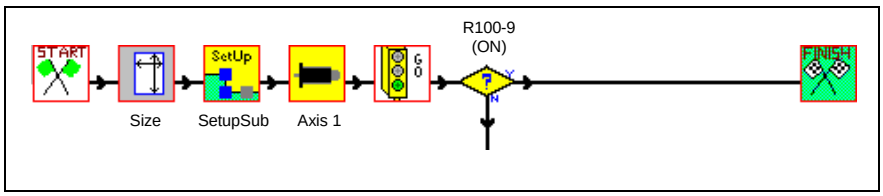
The following variables are used in this program and each one has been assigned a corresponding label:

F1 - Return_Position	I1 - Timer_1 (Return Dwell)
F2 - Forward_Position	I2 - Timer_2 (Forward Dwell)
F3 - Return_Speed	I3 - Part_Counter
F4 - Forward_Speed	
F5 - Accel_Rate	

The dash lines are used as references to the subroutines that are called by the *SetUp* icon Setup*Sub* in the main “Task A” program. To view these subroutines select “Subroutines” from the View menu. To return to the main program select “Task A” from the View menu.

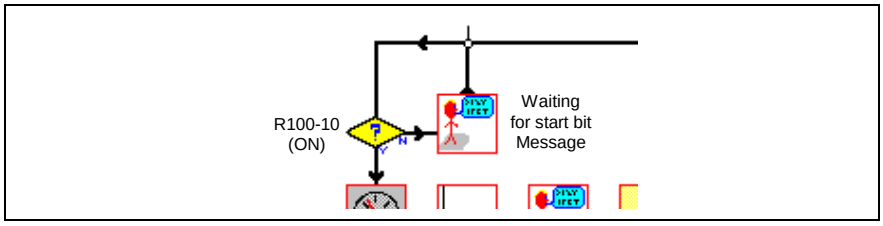


Program Setup



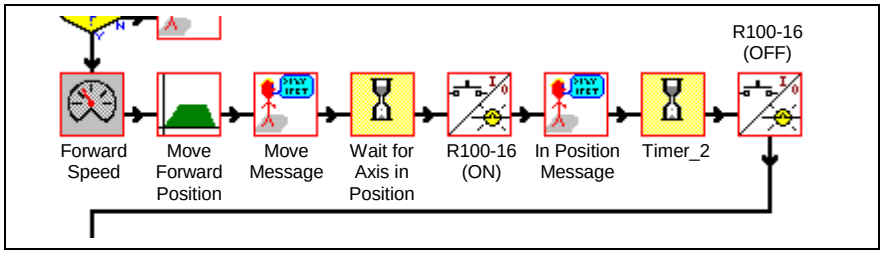
This section starts the program, allocates memory space for variables and sets up Axis 1. A setup subroutine is then executed (see below). The Go icon enables Axis 1 for non-coordinated motion. The branch icon continues the program if Register 100 bit 9 = 0. If the bit is high (1), the program will end.

Start / Stop Loop



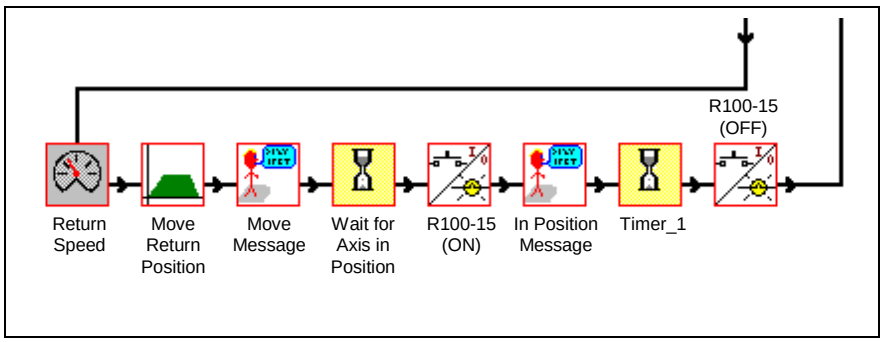
This section starts and stops Forward and Return motion dependent on Register 100 bit 10. If bit 10 is off the program will loop through the "Waiting for Start Bit" message icon. If bit 10 is on, the program will continue down to the forward and return move sections. After each cycle the program returns to this branch, adds 1 to the part count and then checks the status of bit 10 again.

Move Forward Position



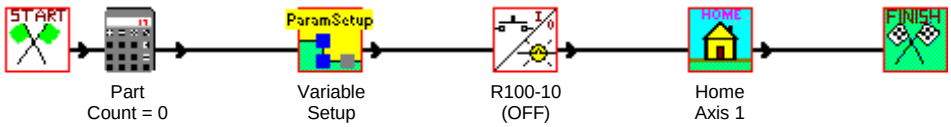
This section sends the velocity rate value (*Forward_Speed*) and move distance value (*Forward_Position*) to the drive and initiates the forward position move. The message icon sends a "Moving to forward position" status message. The wait icon suspends further program execution until the axis has reached the forward position. Once in the forward position Register 100 bit 16 turns on the "in position" indicator light on the interface. The second wait icon is used to cause the forward dwell (*Timer_2*). After the dwell, Register 100 bit 16 is turned off (reset to 0).

Move Return Position



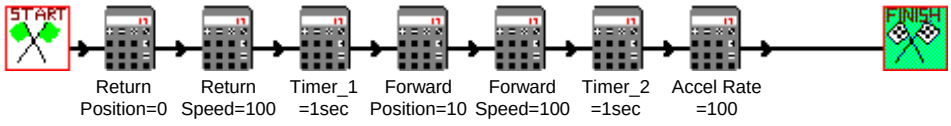
This section sends the velocity rate value (*Return_Speed*) and move distance value (*Return_Position*) to the drive and initiates the reverse position move. The message icon sends a “Moving to return position” status message. The wait icon suspends further program execution until the axis has reached the reverse position. Once in the return position Register 100 bit 15 turns on the “in position” indicator light on the interface. The second wait icon is used to cause the return dwell (*Timer_1*). After the dwell, Register 100 bit 15 is turned off (reset to 0).

Setup Subroutine (Counter Reset)



This subroutine resets the part counter to zero and runs the following Variable Setup subroutine. It moves Axis 1 to its home position. After running the subroutine the program returns back to Task A.

Variable Setup Subroutine



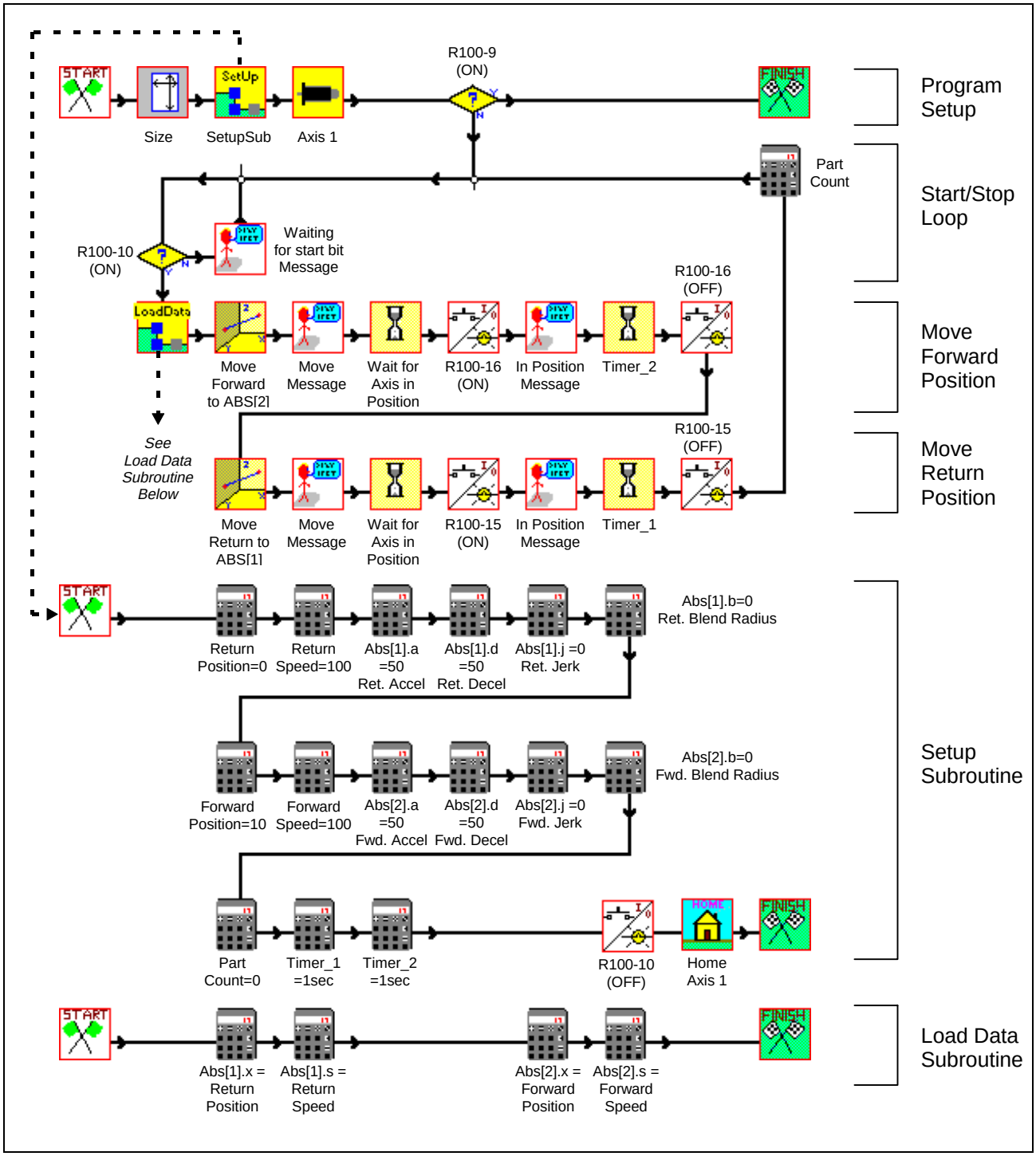
This subroutine sets the variable values for all Floats and Integers. The user can make changes to these variables by selecting ***Data*** ⇒ ***Variables***.

F1 - Return_Position
F2 - Forward_Position
F3 - Return_Speed
F4 - Forward_Speed
F5 - Accel_Rate

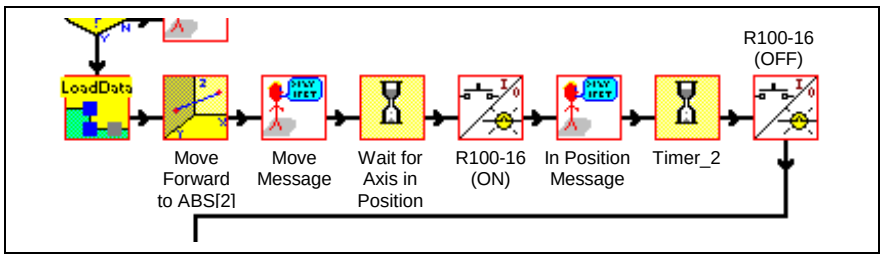
I1 - Timer_1 (Return Dwell)
I2 - Timer_2 (Forward Dwell)
I3 - Part_Counter

VisualMotion Icon Language Program - Demo_2 (Coordinated Motion)

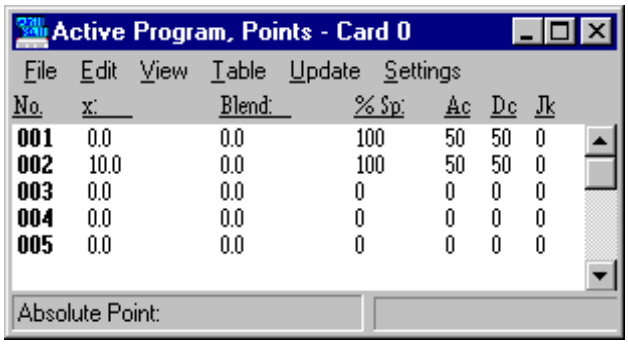
Demo_2 is a Coordinated Motion program that is similar to Demo_1 with the exception of movement type. Demo_2 uses a series of points within a absolute points table that contains the values required for motion. The starting and ending of Demo_2 is identical to that of Demo_1.



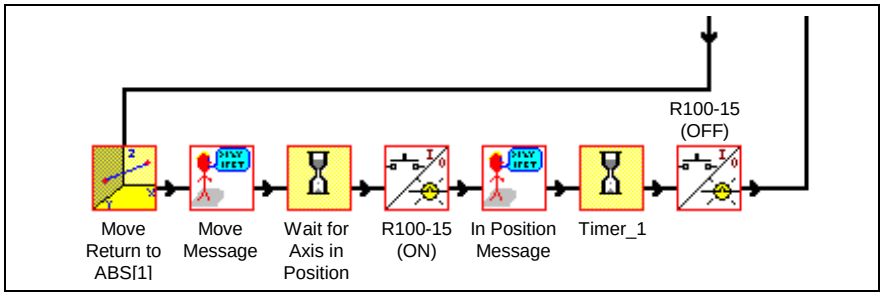
Move Forward Position



Demo_2 uses coordinated motion so the Velocity and Move icons used in Demo_1 have been replaced with a Load Data subroutine and a Path icon. The Load Data Subroutine (See below) assigns values to the absolute point table.

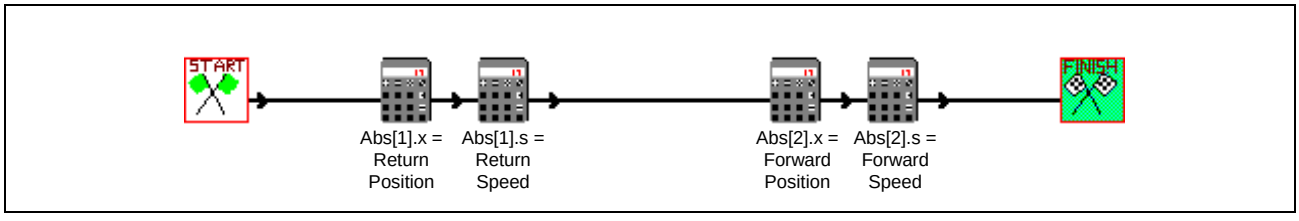

Active Program, Points - Card 0
File Edit View Table Update Settings
No. x: Blend: % Sp: Ac Dc Jk
001 0.0 0.0 100 50 50 0
002 10.0 0.0 100 50 50 0
003 0.0 0.0 0 0 0 0
004 0.0 0.0 0 0 0 0
005 0.0 0.0 0 0 0 0
Absolute Point:

The Path Icon uses these values to set up coordinated absolute motion. An absolute move begins from the current position (or the endpoint of the previous path segment) and terminates at the absolute point specified. In this case the forward position would be ABS[2]. Speed, acceleration and deceleration values are also provided with each point move.



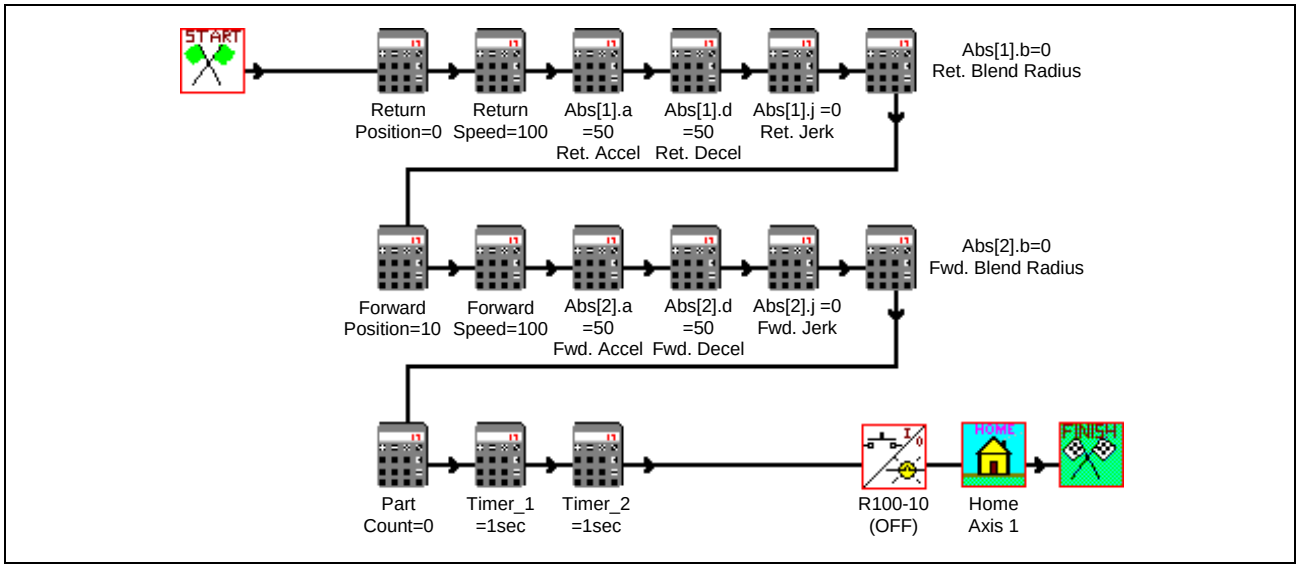
The return position is ABS[1], so the return move will begin at ABS[2] and then end at ABS [1].

Load Data Subroutine



This subroutine associates the variables F1-F4 (Return/Forward Position and Speed) to the absolute point table (Abs[1].x, Abs[2].x, Abs[1].s, Abs[1].x).

Variable and Point Table Setup Subroutine



The variable and point table subroutine assigns values to the association made within the Load Data Subroutine for Forward/Return Position and Speed. This subroutine also adds values to the absolute point table for axis acceleration (Abs[#].a), deceleration (Abs[#].d), jerk(Abs[#].j, and blend radius (Abs[#].b). In addition, the axis is homed and register 100 bit 10 is turned off.

I Index

A

Absolute point table D-24, D-43
acceleration profiles 1-13
Add User Label ***3-12***
Archiving VisualMotion System 2-18

B

Boolean equation 4-4
BTC06 1-10

C

Cam
- coefficient D-21
- coefficients D-24
- equation D-21
- Phase Adjust D-26
- Spline D-24
- table D-21, D-24

Cam synchronization **1-13**
CLC
- Operating System 1-10
- System Architecture 1-9

CLC Card
- Installation and Setup 2-3

CLC-D Card
- configuration 2-5
- installation 2-5

CLC-D Installation Screens 2-34
CLC-D Memory Reset **2-6**
CLC-P01 Card
- configuration 2-8
- installation 2-8

CLC-P01 card number and base memory selection 2-10
CLC-P01 Installation 2-13
CLC-P01 Installation Screens .. 2-35
CLC-P01 Jumper Location 2-11
CLC-P01 memory allocation 2-10
CLC-P01 Memory Reset **2-9**
CLC-P02 Card
- configuration 2-14
- installation 2-14

CLC-P02 Flash Firmware Upgrade 2-18
CLC-P02 Installation 2-17
CLC-P02 Installation Screens .. 2-36
CLC-P02 Jumper Locations 2-15
CLC-V
- Mode Select 2-21

CLC-V Address Switch 2-22
CLC-V Card
- Backplane Communication B-4
  - Direct Data Access B-5
  - Memory Access Limitations B-5
- configuration 2-21
- installation 2-21
- Memory B-1

CLC-V Flash Firmware 2-25
CLC-V Installation 2-24
CLC-V Installation Screens 2-33
coordinated motion D-43

D

Data Menu
- I/O Mapper 4-3
- Variables 3-15

DDE 1-10
DDE client interfaces 5-1
DDE request 5-1
DDEInitiat ***5-4***
DDEInitiate *5-6*
DDEPoke *5-4*
DDERequest *5-4*
DEA card 4-8
DEA I/O C-4
DIAX03 digital drive 1-4
DIAX03 digital drives 2-2
DIAX03 Motors 1-5
DIAX04 digital drives 2-2
DIAX04 drive family 1-6
DIAX04 Motors 1-7
DPRAM register A-1
Drive Homing 3-4
Drive Programming Module 2-2
DSS drive address settings 2-4
DSS SERCOS communication card 2-3

E

ECODRIVE 3 digital drives 2-2
ECODRIVE 3 PLS Function D-34
ECODRIVE drive family 1-8
ELS
- master D-30
- slave D-30
- slave axis D-21
- virtual master D-21

EPROM 2-25
Erasing CLC-V User Programs from Flash 2-28
Event functions 3-11
events 1-10, 1-11

F

Fiber Optic Cables 2-2
Fiber optic ring structure 2-6
File Extensions **3-16**
File Extensions
- .acc 3-16
- .exb 3-16
- .exc 3-16
- .iom 3-16
- .lss 3-16
- .lst 3-16
- .map 3-16
- .pos 3-16
- .prm 3-16
- .str 3-16
- .tbl 3-16
- .txt 3-16
- .vel 3-16
- csv 3-16

File Types 3-16
floating point variable 3-15
function argument D-1

G

Global variables 3-15

H

Hardware Requirements 2-1
Host System
  requirements 2-1

I

I/O D-1
I/O Mapper 4-3
  Display Strings 4-4
  Uploading I/O Mapper 4-3
Icon
  branch D-18
  Cam Build: D-24
  I/O D-18
  Sequencer D-18
Icons
  Start 3-1
Integers 3-15
Item name **5-2**, *5-4*

L

Labels 3-12
Local variables 3-15

M

Modifying a Parameter List D-35
motion
  coordinated 1-12
    circular interpolation 1-13
    constant speed 1-12
    kinematics 1-13
    linear interpolation 1-13
  ELS 1-13
    phase synchronous mode 1-13
    velocity synchronous mode 1-13
  non-coordinated 1-12
    ratioed axes 1-12
    single axis 1-12
    velocity mode 1-12
Multi-turn Encoder 3-5

O

OPERATOR
  DEVICES/INTERFACES 5-1

P

Parameter
  Overview 4-12
PC Backplane Communications . 5-9
PC/104 Base Address selection
  (CLC-P02) 2-15
PC/104 Interrupt selection 2-11, 2-16
Pentland MPV922 ..... C-22
Phase synchronization **1-13**
PLS D-30
pROBE function 2-18, 2-25
program execution 4-1
Programmable Limit Switch (PLS) D-30
Programming PLS ON/OFF Signal D-37
  P-0-0131 ..... D-37
  P-0-0132 ..... D-37
  P-0-0133 ..... D-37
  P-0-0134 ..... D-37
  P-0-0135 ..... D-37

R

Restoring CLC-V User Programs
  from Flash 2-27

S

S-0-0026 ..... D-34
S-0-0144 ..... D-34
S-0-0328 ..... D-34
S1 Dip Switch 2-15
sample program 3-1
Saving CLC-V User Programs to
  Flash 2-27
Scmd D-21
Sequencer
  Application D-1
  editor D-1
  list D-1
  step D-1
SERCOS 1-9, 2-2, 2-7
Serial Cables 2-2
Serial Communications 5-9
Serial I/O 2-1
Service name **5-2**
Setup Types 2-31
Signal Status Word D-34
Signal Status Word Configuration D-36
slave axis D-21
Subroutine 3-11

T

Tagnames 5-10
Task 3-11
tasks 1-10
Topic name **5-2**, *5-9*

V

Variables 3-11, ***3-15***
Velocity synchronization **1-13**
View Menu
  Event functions 3-11
  Subroutine 3-11
  Task 3-11
VisualMotion 1-10
VisualMotion Overview 1-3
VisualMotion system 1-3
***VisualMotion Toolkit installation
  and Setup 2-12***, 2-16, 2-23, 2-29
VM System Motion Capabilities 1-12
VME 2-22
VME Backplane Communications 5-9
VME Bus request level 2-22
VME Card Cage C-1
  Backplane Jumpers C-3
  Input Power C-3
  Secondary Battery C-1
    Ordering C-2
VME Configuration B-7
  Address and Access Modes B-7
  System Signals B-8
    Bus Arbitration B-8

Bus Interface B-9
Bus Release Modes B-9
VME I/O Systems C-4

W

Windows 95/98 memory allocation2-9
Windows NT Base memory setup2-11

X

Xycom XVME-201..... C-8
Xycom XVME-202..... C-14
Xycom XVME-244..... C-18

engineering
mannesmann
Rexroth

Supplement A
Profibus Fieldbus Interfaces
for VisualMotion 6.0

Indramat

Contents

1 General Information 1-1

1.1 CLC-D System Description with a Fieldbus 1-1
The VisualMotion Fieldbus Mapper 1-1
1.2 Data Transfer Direction (Output vs. Input) 1-2
1.3 Fieldbus Data Channel Descriptions 1-2
Cyclic Channel 1-2
The Real-Time Channel 1-3
Parameter Channel (Profibus DPF05.x with CLC-D only) 1-6
Non-Cyclic Channel 1-7
Direct-Mapped Data 1-7
Data Exchange Objects 1-8

2 Fieldbus Mapper Examples 2-1

2.1 Basic Example 2-1
STEP I: Determine the Cyclic and Non-Cyclic Data 2-1
STEP II: Configure Fieldbus Slave Card with Setup Information and Cyclic Data Object Lists Using the Fieldbus Mapper 2-2
STEP III: Define Cyclic Data Mapping Lists 2-5
STEP IV: Define Non-Cyclic Data Mapping Lists (Direct Mapping) 2-8
2.2 Multiplexing Example 2-11
STEP I: Determining the Cyclic and Non-Cyclic Data (with Multiplexing) 2-11
STEP II: Configure Fieldbus Slave Card with Setup Information and Cyclic Data Object Lists Using the Fieldbus Mapper 2-12
STEP III: Define Cyclic Data Mapping Lists 2-16
STEP IV: Define Non-Cyclic Data Mapping Lists (Direct Mapping) 2-18
2.3 Parameter Channel Example 2-20

3 Information for the GPS Programmer 3-1

3.1 Viewing/Printing a Summary Report of the Current Fieldbus (only when CLC-D is on-line) 3-1
To View a Summary Report: 3-1
To Print a Summary Report: 3-2
3.2 Fieldbus-Accessible Parameters 3-3
3.3 Register 19 Definition (Fieldbus Status) 3-6
Diagnostic Object 5FF2 3-6
Bit Definitions 3-6
3.4 Register 20 Definition (Fieldbus Diagnostics) 3-7
Diagnostic Object 5FF0 3-7
Bit Definitions 3-8
3.5 Register 26 Definition (Fieldbus Resource Monitor) 3-8

Bit Definitions 3-9
3.6 Fieldbus Error Reaction 3-9

4 Information for the PLC Programmer 4-1

4.1 *.gsd File 4-1
4.2 Multiplex Data Bits in the Control and Status Words 4-1
4.3 Parameter Channel in the Cyclic Data Channel (Process Data Channel on Profibus DPF05.x only) 4-3
P-CHAN (Parameter Channel) Component Descriptions 4-4
Bit Definitions of the Fieldbus Control / Status Word: 4-4
Messaging Formats 4-6
Short Format 2: General Explanation 4-6
Short Format 2: Structure of the Command Telegram 4-7
Short Format 2: Structure of the Response Telegram 4-7
Short Format 2: Examples 4-8
VisualMotion ASCII Format: General Explanation 4-11
VisualMotion ASCII Format: Structure of the Command Telegram 4-11
VisualMotion ASCII Format: Structure of the Response Telegram 4-12
VisualMotion ASCII Format: Example 4-13
4.4 Non-Cyclic Transmission (Direct-Mapped Objects) 4-16
Selecting a Direct-Mapped Object 4-16
Transmission Sequence via a Direct-Mapped Object 4-16
Non-Cyclic Direct-Mapped Write 4-17
Non-Cyclic Direct-Mapped Read 4-18
4.5 Non-Cyclic Transmission (Data Exchange Objects) 4-19
Selecting a Data Exchange Object 4-19
Transmission Sequence via a Data Exchange Object 4-19

5 Profibus DP Combi Slave Board DPF05.x 5-1

5.1 Application 5-1
5.2 Functional Description 5-1
5.3 Profibus Interface 5-1
5.4 DPF05.x Board Hardware 5-2
Front view of the DPF05.x 5-2
DPF05.x Structure 5-2
X68 Connector, Profibus Pinouts 5-3
X69 Connector, External Inputs 5-3
DPF05.x Diagnostics 5-3
Front Panel LEDs 5-3
Definition of Diagnostic LEDs 5-4

6 Index 6-1

1 General Information

Important: Using a Fieldbus with the VisualMotion system will consume additional processing resources on the CLC-D card. Please take into consideration the consumption of these resources, especially when adding a Fieldbus to an existing application.

Version Note:
Information in this document is based on VisualMotion Toolkit software version 06V12 and CLC-D firmware version GPS06V64.

1.1 CLC-D System Description with a Fieldbus

The CLC-D can operate on a serial Fieldbus interface (network) by means of a plug-in card (DPF05.x board) that communicates with the CLC-D card via dual-port RAM. The function of the Fieldbus card is similar to that of a network card in a PC: it allows communication with other devices on the network.

In ***Figure 1-1: Sample Master/Slave Setup with Fieldbus Cards***, a commonly described Fieldbus interface is pictured:

- **Fieldbus Master** - PLC Fieldbus interface
- **Fieldbus Slave** - CLC-D Fieldbus interface

In this document, we will refer to the PLC as the **Fieldbus Master** and the CLC-D as the **Fieldbus Slave**.

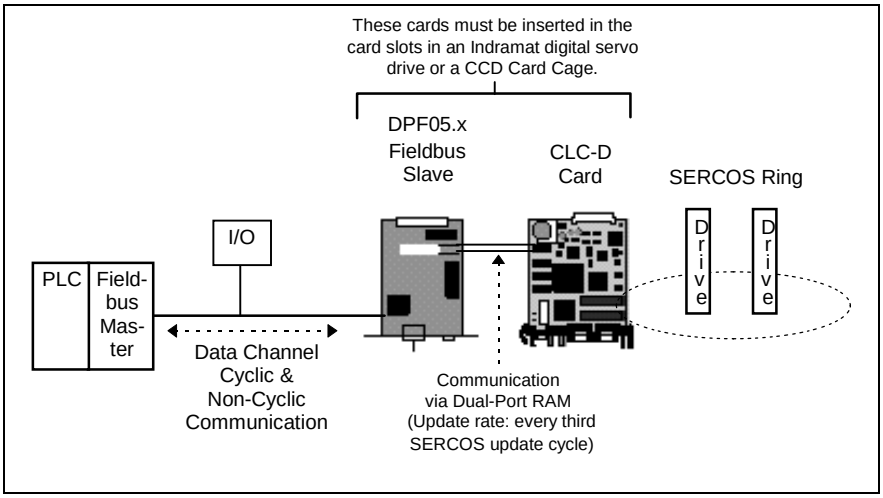

These cards must be inserted in the card slots in an Indramat digital servo drive or a CCD Card Cage.
DPF05.x Fieldbus Slave
CLC-D Card
SERCOS Ring
I/O
PLC
Field-bus Mas-ter
Drive
Drive
Data Channel Cyclic & Non-Cyclic Communication
Communication via Dual-Port RAM (Update rate: every third SERCOS update cycle)

Figure 1-1: Sample Master/Slave Setup with Fieldbus Cards

With the CLC-D card, the Profibus (DPF 05.x) Fieldbus card can be used **only** as a **slave** card in a master/slave setup.

Important: When using a Fieldbus slave interface, it is recommended to use a 4 ms or higher SERCOS update.

The VisualMotion Fieldbus Mapper

In the VisualMotion software package, the Fieldbus Mapper is a tool used to set up Fieldbus data. It is primarily an on-line tool, used most effectively when connected to a CLC-D card.

Important: Whenever the Fieldbus Mapper is configured on-line, the system must be in Parameter Mode.

1.2 Data Transfer Direction (Output vs. Input)

In the VisualMotion Fieldbus Mapper, output and input are always described with respect to the Fieldbus master (the Fieldbus card associated with the PLC). The definitions for output and input follow:

output: the communication from the PLC to the CLC-D card (i.e. from the Fieldbus master to the Fieldbus slave).
Synonyms for this type of communication: **send** or **write** data.

input: the communication from the CLC-D card to the PLC (i.e. from the Fieldbus slave to the Fieldbus master).
Synonyms for this type of communication: **receive** or **read** data.

1.3 Fieldbus Data Channel Descriptions

The Indramat Profibus Fieldbus interface card for the CLC-D (DPF card) supports two data channels for input/output communication with the CLC-D:

1. **Cyclic Channel:** DP (Decentralized Peripheral)
   - ***Real-Time Channel*** (for **single** and **multiplex** channels)
   - ***Parameter Channel*** (for systems requiring non-cyclic transmissions, but without FMS support from the master)
2. **Non-Cyclic Channel:** FMS (Fieldbus Message Specification)

Cyclic Channel

The DPF05.x Fieldbus card has pre-defined cyclic bus objects (16- or 32-bit) which are declared and transmitted as an ordered list (the bus configuration list). This bus configuration list acts as a placeholder for CLC-D mapped data from the CLC-D object mapping list. See *Object Lists and Their Transfer Locations.*

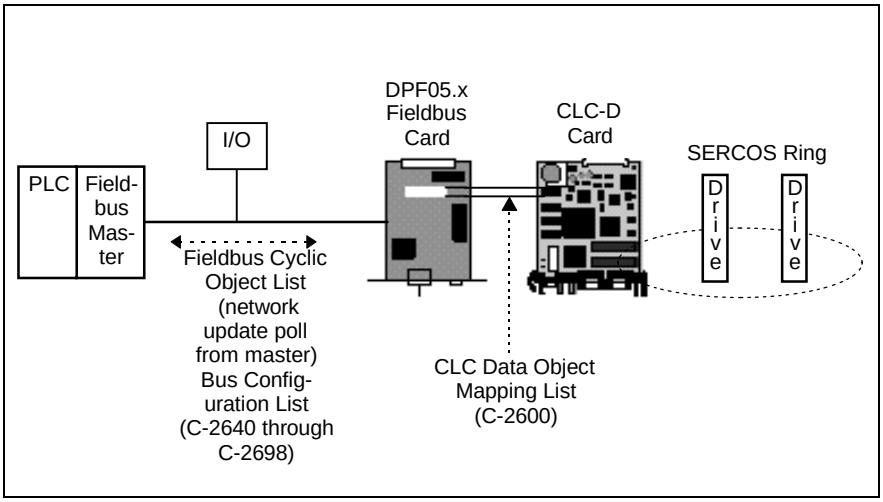


Figure 1-2: Object Lists and Their Transfer Locations for Cyclic Data

The cyclic data channel is limited to a total of 16 input words and 16 output words. CLC-D data types consume these objects in either one-word (or 16-bit) groups for CLC registers or two-word (or 32-bit) groups for all other data types in the object mapping list (C-0-2600).

Note: The bus configuration list must be set up in the field bus mapper bus configuration screen before the object mapping list can be configured.

For the cyclic data, the CLC-D data object mapping list is scanned every third SERCOS update cycle and data is sent and received to/from the slave board's dual port RAM.

The cyclic data channel can be made up of any combination of the following data types:

- Real-Time Channel
  - Single Channel
  - Multiplex Channel
- Parameter Channel

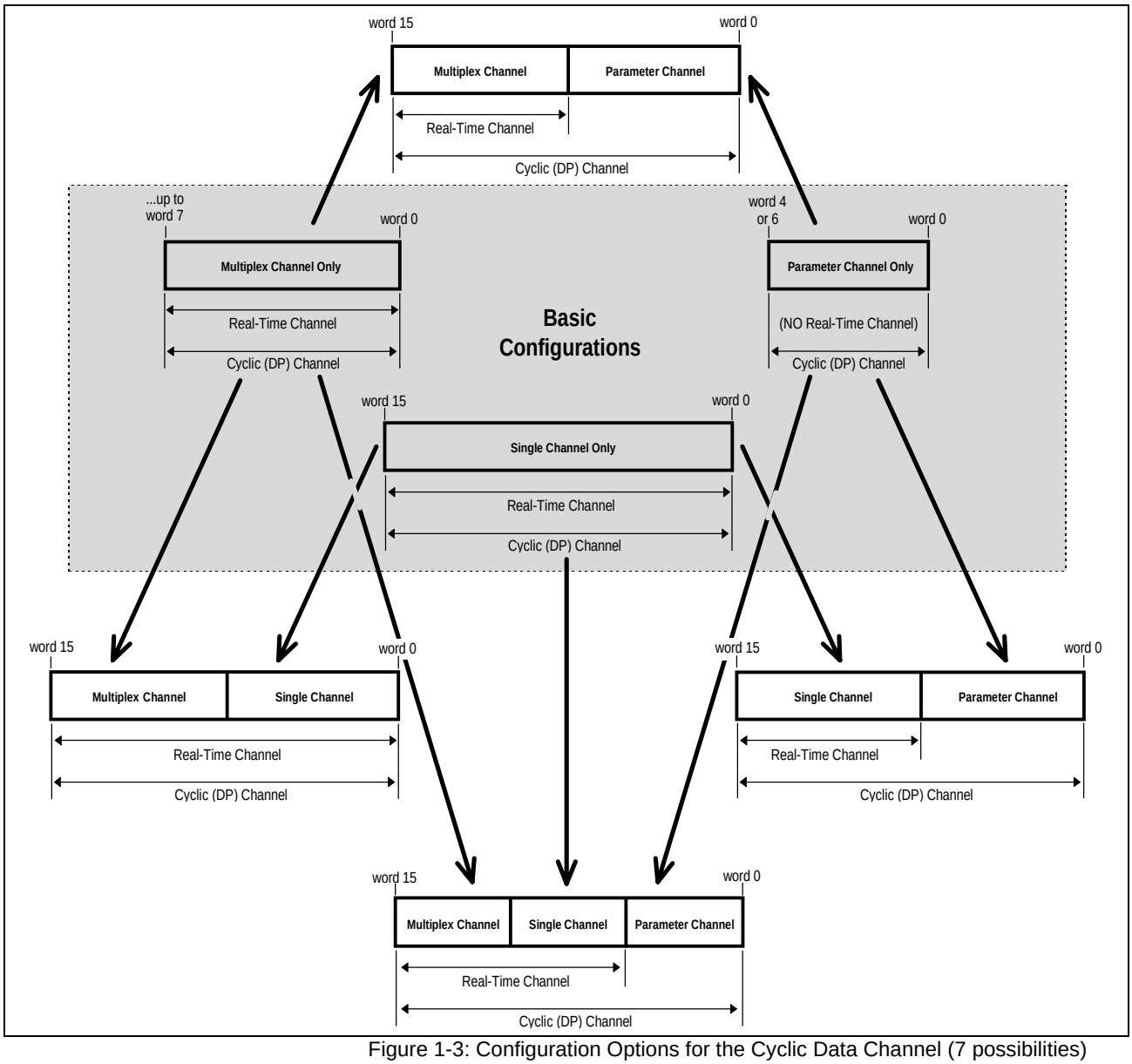


Figure 1-3: Configuration Options for the Cyclic Data Channel (7 possibilities)

The Real-Time Channel

In the real-time channel, data is updated cyclically between the Fieldbus master and slave. This channel contains two possible data types: **single** and **multiplex**.

Data Objects: Types and Sizes

The following table outlines the CLC-D data object types that can be transmitted via the cyclic channel and the amount of space (in 16-bit data words) that each data type consumes. Remember, the cyclic channel is limited to 16 data words in each direction (input and output).

Note: The cyclic data mapping list supports only 16- and 32-bit data of the following types for reading and writing:
- Integer
- Float
- Binary (used in CLC parameters)
- Hex (used in CLC parameters)

For all other data types (e.g. diagnostic messages - “strings”), use the non-cyclic Data Exchange Object or the Parameter Channel.

CLC Data Type	Data Size (in Data Words)
Register	1
Integer (currently active program ONLY *)	2
Float (currently active program ONLY *)	2
Global Integer	2
Global Float	2
CLC Card Parameter	2
CLC Axis Parameter	2
CLC Task Parameter	2
Note: Drive parameters "S" or "P" cannot be transmitted cyclically because of the inherent delay of parameter access over the SERCOS service channel. See ***"Non-Cyclic Channel/Data Exchange Objects"*** on page 1-8. However, if a drive parameter is mapped to an axis parameter, that axis parameter could be used in cyclic data (see description of Axis Parameters 180-196 in the **VisualMotion 6.0 Reference Manual**).	
*** Important Note:** Integers and floats are shown only for the currently active program. Each time you activate a new progam, the fieldbus reads/writes to the newly-activated program.	

Table 1-1: CLC-D Cyclic Data Types and Sizes

Single Data Types

Indramat Fieldbus interfaces have single (16-bit) and double (32-bit) word objects in the cyclic data channel with which simple applications can be handled without any difficulty. These objects are directly mapped to CLC-D data types and updated every third SERCOS update cycle.

Multiplex Data Types (Cyclic Data Channel)

In some multi-axis applications, 16 words of cyclic data transfer are not sufficient to meet the data transfer requirement of the application.

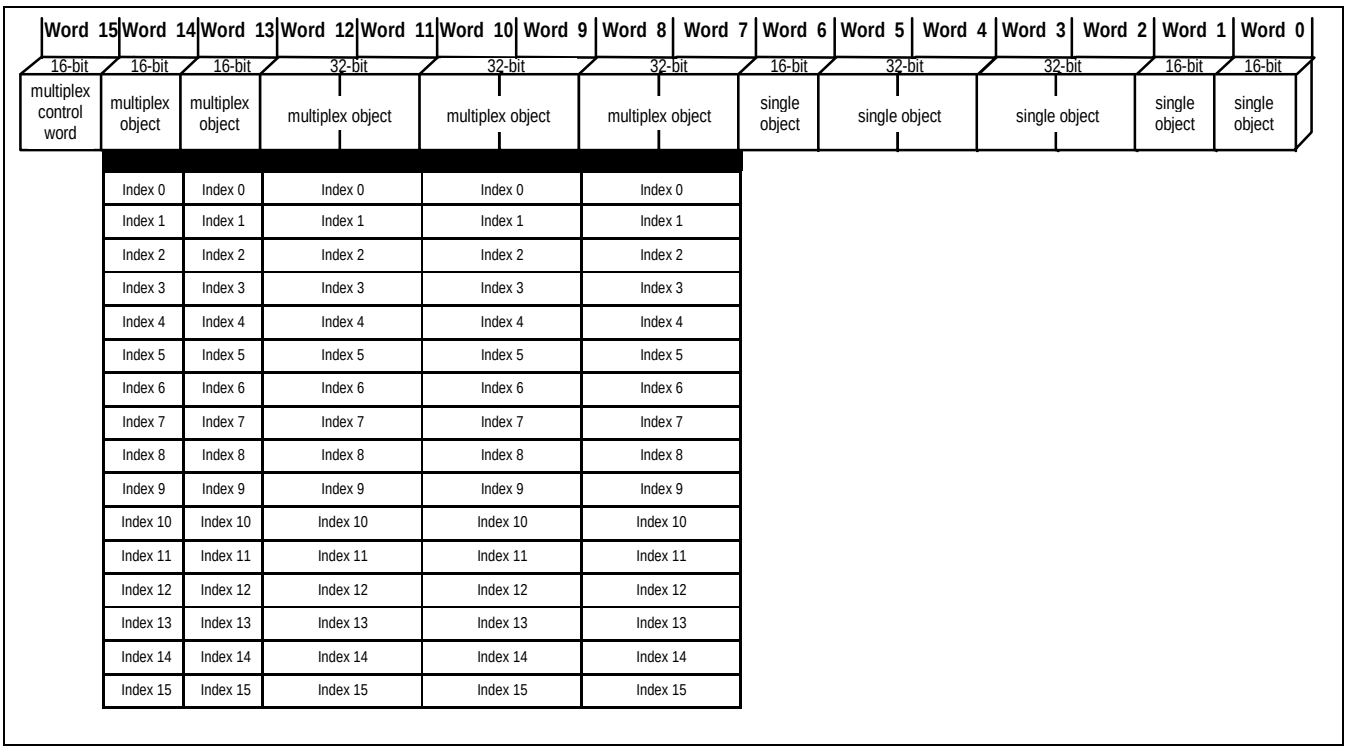
When insufficient data transfer space is available, multiplex data can be set up within the cyclic channel. One data object (base object) acts as a placeholder for multiple possible CLC-D data types (all of the same word size). The currently transmitted CLC-D data type is based on an index value placed in a multiplex control word attached to the end of the cyclic data list. Depending on the index specified by the master, the multiplex channel permits a different set of data within the cyclic channel to be transferred as current real-time data in both data directions.

Note: Using the multiplex channel will reduce the maximum number of usable words for storing CLC data to 15. The 16th word (or last used word, if fewer than 15 words) is used as the multiplex entry control/status word.

In the Fieldbus configuration lists, the following multiplex objects are available:

Type of Multiplex Object	Input Objects Available	Total Input Objects	Output Objects Available	Total Output Objects
32-bit	3 base (x16 indices)	48	3 base (x16 indices)	48
16-bit	2 base (x16 indices)	32	2 base (x16 indices)	32

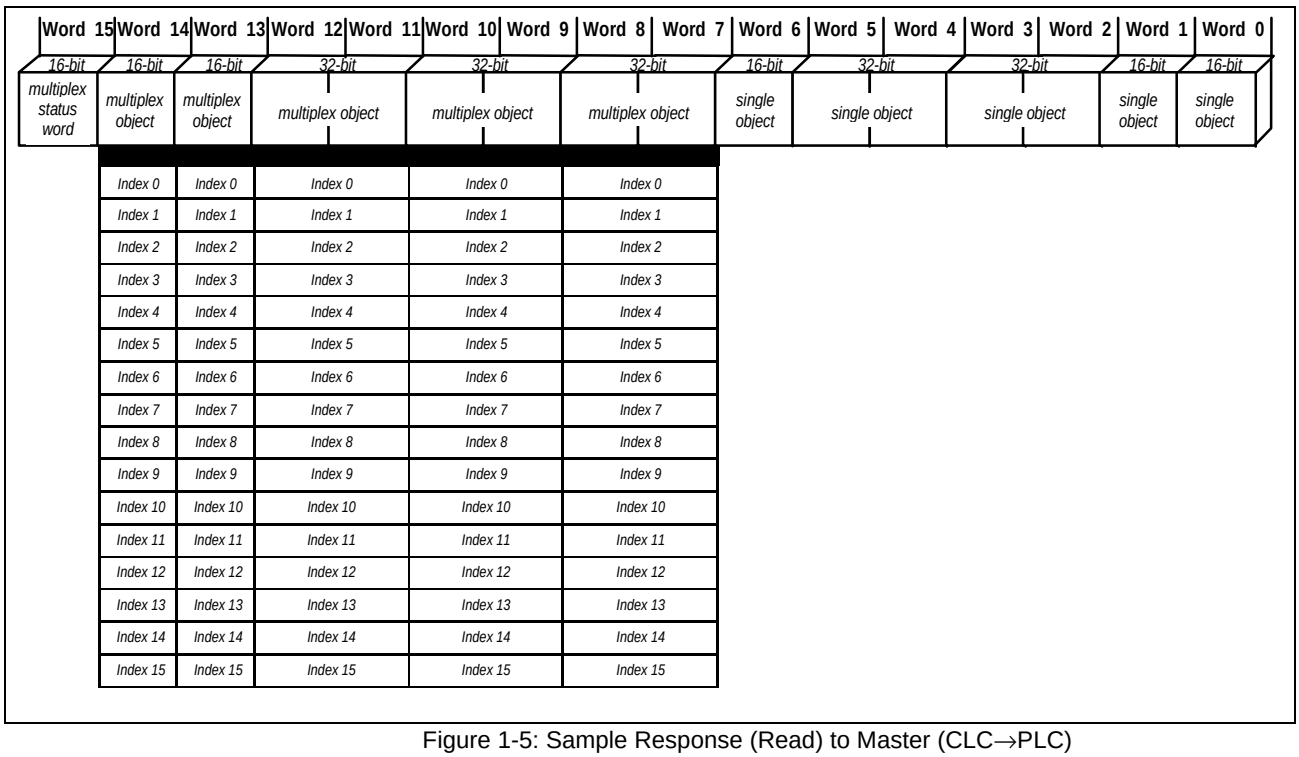
Table 1-2: Available Multiplex Objects



Word 15	Word 14	Word 13	Word 12	Word 11	Word 10	Word 9	Word 8	Word 7	Word 6	Word 5	Word 4	Word 3	Word 2	Word 1	Word 0
16-bit	16-bit	16-bit	32-bit		32-bit		32-bit		16-bit	32-bit		32-bit		16-bit	16-bit
multiplex control word	multiplex object	multiplex object	multiplex object		multiplex object		multiplex object		single object	single object		single object		single object	single object

Index 0	Index 0	Index 0	Index 0	Index 0
Index 1	Index 1	Index 1	Index 1	Index 1
Index 2	Index 2	Index 2	Index 2	Index 2
Index 3	Index 3	Index 3	Index 3	Index 3
Index 4	Index 4	Index 4	Index 4	Index 4
Index 5	Index 5	Index 5	Index 5	Index 5
Index 6	Index 6	Index 6	Index 6	Index 6
Index 7	Index 7	Index 7	Index 7	Index 7
Index 8	Index 8	Index 8	Index 8	Index 8
Index 9	Index 9	Index 9	Index 9	Index 9
Index 10	Index 10	Index 10	Index 10	Index 10
Index 11	Index 11	Index 11	Index 11	Index 11
Index 12	Index 12	Index 12	Index 12	Index 12
Index 13	Index 13	Index 13	Index 13	Index 13
Index 14	Index 14	Index 14	Index 14	Index 14
Index 15	Index 15	Index 15	Index 15	Index 15

Figure 1-4: Sample Command (Write) to Slave (PLC→CLC)



Word 15	Word 14	Word 13	Word 12	Word 11	Word 10	Word 9	Word 8	Word 7	Word 6	Word 5	Word 4	Word 3	Word 2	Word 1	Word 0
16-bit	*16-bit*	*16-bit*	*32-bit*		*32-bit*		*32-bit*		*16-bit*	*32-bit*		*32-bit*		*16-bit*	*16-bit*
multiplex status word	*multiplex object*	*multiplex object*	*multiplex object*		*multiplex object*		*multiplex object*		*single object*	*single object*		*single object*		*single object*	*single object*

Index 0	*Index 0*	*Index 0*	*Index 0*	*Index 0*
Index 1	*Index 1*	*Index 1*	*Index 1*	*Index 1*
Index 2	*Index 2*	*Index 2*	*Index 2*	*Index 2*
Index 3	*Index 3*	*Index 3*	*Index 3*	*Index 3*
Index 4	*Index 4*	*Index 4*	*Index 4*	*Index 4*
Index 5	*Index 5*	*Index 5*	*Index 5*	*Index 5*
Index 6	*Index 6*	*Index 6*	*Index 6*	*Index 6*
Index 7	*Index 7*	*Index 7*	*Index 7*	*Index 7*
Index 8	*Index 8*	*Index 8*	*Index 8*	*Index 8*
Index 9	*Index 9*	*Index 9*	*Index 9*	*Index 9*
Index 10	*Index 10*	*Index 10*	*Index 10*	*Index 10*
Index 11	*Index 11*	*Index 11*	*Index 11*	*Index 11*
Index 12	*Index 12*	*Index 12*	*Index 12*	*Index 12*
Index 13	*Index 13*	*Index 13*	*Index 13*	*Index 13*
Index 14	*Index 14*	*Index 14*	*Index 14*	*Index 14*
Index 15	*Index 15*	*Index 15*	*Index 15*	*Index 15*

Figure 1-5: Sample Response (Read) to Master (CLC→PLC)

Multiplex Control and Status Words

The Profibus multiplex control and status words serve to command and acknowledge multiplex data transferred between the Fieldbus master and the Fieldbus slave. The **control** word is associated with **output** communication (PLC→CLC). The **status** word is associated with **input** communication (CLC→PLC). Single data objects are not affected by the multiplex control and status words.

Note: For specific information about how the Fieldbus master uses the multiplex control and status words, see ***Multiplex Data Bits in the Control and Status Words*** on page 4-1.

Sending and Receiving the Same CLC-D Data Cyclically

Two separate objects will be consumed when sending and receiving the same CLC-D data (both cyclically and non-cyclically).

If you wish to cyclically send and receive the same CLC-D data, the Output mappings should come **first** in the list (see ***Fieldbus Mapper Examples, Basic Example*** on page 2-1).

Parameter Channel (Profibus DPF05.x with CLC-D only)

For Profibus systems using the CLC-D/DPF05.x/VisualMotion configuration, where support of non-cyclic (FMS) communication is not available over the Fieldbus, a subset of the cyclic (DP) channel can be allocated for non-cyclic communications (e.g. parameterization and extended diagnostic information). This subset of the cyclic channel is called the **parameter channel**.

Note: The parameter channel is always allocated as the first 4 or 6 words of the Profibus cyclic (DP) channel. The length of the parameter channel plus the length of the real-time channel used to exchange cyclic data represent the entire length of the DP channel (maximum total length: 16 words). See ***Figure 1-3: Configuration Options for the Cyclic Data Channel (7 possibilities)*** for DP channel configuration options. For an example of how to set up the parameter channel, see ***Parameter Channel Example*** on page 2-20.

Two messaging formats are available in the parameter channel, to allow for a varying degree of implementation, depending on application requirements:

- **Short Format 2-** simple messaging format for basic, limited requirements
- **VisualMotion ASCII Format**- complex messaging format, when flexibility and extensive diagnostics are required

The data in the message header (the parameter channel control or status word) determines which message type is being sent or received.

Short Format 2

This format allows for direct access to the Fieldbus objects on the DPF board. (These objects can also be directly mapped to VisualMotion data types.) Short Format 2 is the easiest format to implement from the PLC side, but it is also the most limiting in the flexibility of access and the number of different VisualMotion types. Setup is the same as for non-cyclic objects (see ***Non-Cyclic Data*** on page 2-1 for the setup procedure in the Fieldbus Mapper).

Note: When using **only** Short Format 2 messaging, only 4 words need to be allocated in the Parameter Channel.

VisualMotion ASCII Format

VisualMotion ASCII Format is the Parameter Channel's method of accessing data exchange objects. For an explanation of the data exchange object, see ***Data Exchange Objects*** on page 1-8. For PLC implementation, see ***Information for the PLC Programmer*** on page 4-1.

Non-Cyclic Channel

The non-cyclic channel is used for data that needs to be transferred only once or sporadically, such as:

- the transmission of lists
- parametriziation of axes or programs
- any non-cyclically mapped data

Instead of being updated during each cycle, non-cyclic data is transferred whenever time is available on the Fieldbus. Though any data type can be transferred non-cyclically, diagnostic messages and drive parameters (S and P) **must** be transferred non-cyclically because of the non-cyclic retrieval for drive parameters through SERCOS and the length of the diagnostic messages.

For example: ASCII text in a diagnostic message requires one data word for every two ASCII characters. The non-cyclic channel can transfer only up to 16 data words (or 32 characters) of a diagnostic message at once. This would mean that one diagnostic text message (if 32 characters or more) could consume all the available cyclic data. For this reason, the transfer of diagnostic messages must be made over the non-cyclic (FMS) or Parameter Channel; it is not allowed over the real-time channel.

There are two types of non-cyclic data transmissions for the CLC-D/VisualMotion system:

- data transmitted via the data exchange object
- data mapped directly to CLC-D data types

Non-cyclic data can be accessed via:

- FMS support of the Fieldbus master
- the parameter channel

Direct-Mapped Data

Just as there are pre-defined cyclic data objects for mapping CLC data to the Profibus DP channel, there are also pre-defined non-cyclic data objects (16- and 32-bit) that can be mapped to CLC data types. Non-cyclic objects have the following numeric names:

- **16-bit:** 5F60 - 5F7F and 5F90 - 5FBF
- **32-bit:** 5F00 - 5F1F

This type of data is mapped using the non-cyclic data mapping list (C-0-2700). It provides easy access to non-cyclic data. However, the fixed mapping list is limited to a certain number and types of CLC-D data. The directly-mapped non-cyclic data (take note of size and direction) can be assigned to the following objects:

Number of Objects Available	**Data Size**	**Direction** (reference: Master Fieldbus)	**Numeric Names of Objects**
16	32-bit	in	5F10-5F1F
16	32-bit	out	5F00-5F0F
32	16-bit	in	5F60-5F6F, 5FA0-5FAF
32	16-bit	out	5F70-5F7F, 5Fb0-5FbF

Table 1-3: Non-Cyclic Data Objects

Note: The available CLC data types for directly-mapped non-cyclic data are the same as for directly-mapped cyclic data.

Data Object Types and Sizes

The following table outlines the directly-mapped CLC-D data object types that can be transmitted via the non-cyclic channel and the amount of space (in data words) that each data type consumes.

CLC Data Object Type	Data Size (in Data Words)
Register	1
Integer (currently active program ONLY)	2
Float (currently active program ONLY)	2
Global Integer	2
Global Float	2
CLC Card Parameter	2
CLC Axis Parameter	2
CLC Task Parameter	2

Table 1-4: Data Object Types/Sizes

Sending and Receiving the Same CLC-D Data Non-Cyclically

Two separate objects will be consumed when sending and receiving the same CLC-D data (both cyclically and non-cyclically).

Data Exchange Objects

Four data exchange objects 5E70 to 5E73 are available for the transfer of non-cyclic data. These objects represent fixed data "containers" of varying lengths that transfer the VisualMotion ASCII Protocol to the CLC-D card, in the same way that data is transferred using the VisualMotion ASCII Format in the Parameter Channel. These objects serve as an open-ended possibility to access any VisualMotion data (including cams, diagnostic text, etc.), but more work is required in the master to perform a transmission of this type. For more specific information about these objects, see **Non-Cyclic Transmission (Direct-Mapped Objects)** on page 4-19.

2 Fieldbus Mapper Examples

2.1 Basic Example

The following example demonstrates the process for configuring the mapping of basic data between the CLC-D card and a Profibus Fieldbus.

Important: The Fieldbus Mapper is an on-line configuration tool. The following conditions must be met when performing STEP I through STEP IV below:
- The system must be powered up.
- The CLC-D card must be connected to the system.
- The system must be set to parameter mode.

STEP I: Determine the Cyclic and Non-Cyclic Data

Cyclic Data

In this example, we will assume that you want to cyclically transfer the following data between the CLC-D card and the Fieldbus master in the order listed.

Note: Quantity and type of data varies depending on application.

Output Data (From Bus to CLC Card)	Size (Data Words)	Size (Bits)	Input Data (From CLC Card to Bus)	Size (Data Words)	Size (Bits)
Register 100	1	16	Integer 29	2	32
Float 2	2	32	Register 47	1	16
Global Integer 3	2	32	Register 120	1	16
Register 104	1	16	Global Float 22	2	32
Register 40	1	16	Axis Parameter 1.100	2	32
Integer 4	2	32	Integer 30	2	32
Integer 5	2	32			
TOTAL	11 Data Words			10 Data Words	

Table 2-1: Sample Cyclic Data

Non-Cyclic Data

We will assume that the following data will be transferred non-cyclically for system initialization or setup. Remember, the direct-mapped non-cyclic data is intended for infrequent data transfer.

Output Data (From Bus to CLC Card)	Size (Data Words)	Size (Bits)	Input Data (From CLC Card to Bus)	Size (Data Words)	Size (Bits)
Float 16	2	32	Float 5	2	32
Integer 7	2	32	Integer 8	2	32
Register 101	1	16			

Table 2-2: Sample Non-Cyclic Data

Note: See ***Table 1-3: Non-Cyclic Data Objects*** for data limitations.

Note: For other data types, such as diagnostic text messages, or S and P parameters, use the data exchange object (see ***P-CHAN (Parameter Channel) Component*** Descriptions on page 4-4).

STEP II: Configure Fieldbus Slave Card with Setup Information and Cyclic Data Object Lists Using the Fieldbus Mapper

Important: The drive must be set to Parameter Mode to configure and assign the data mapping for Fieldbuses (STEP II and STEP III).

1. In the VisualMotion "Data" Menu, select "Fieldbus Mapper."

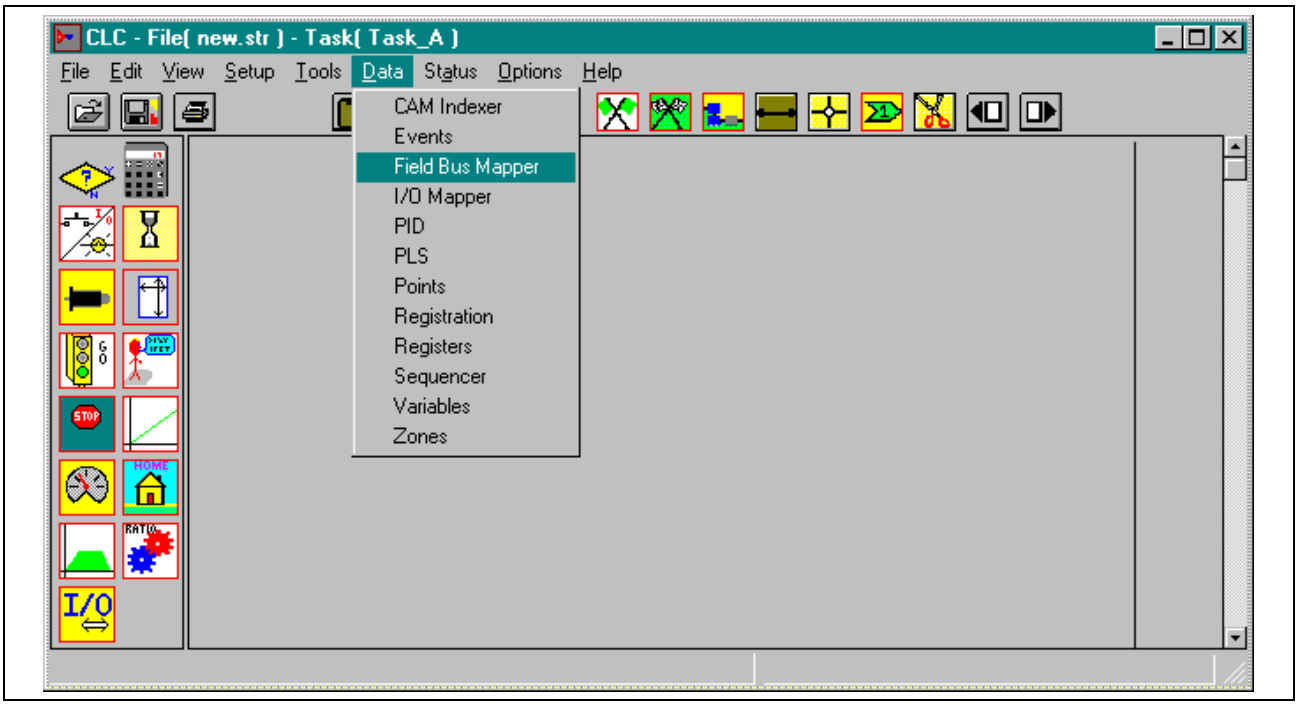


Figure 2-1: Selecting the Fieldbus Mapper from the main VisualMotion Screen

2. In the scroll box under "Bus Type," ensure that the desired bus is selected (If the DPF card is connected, Profibus should appear).
3. Choose the cyclic data channel (DP).

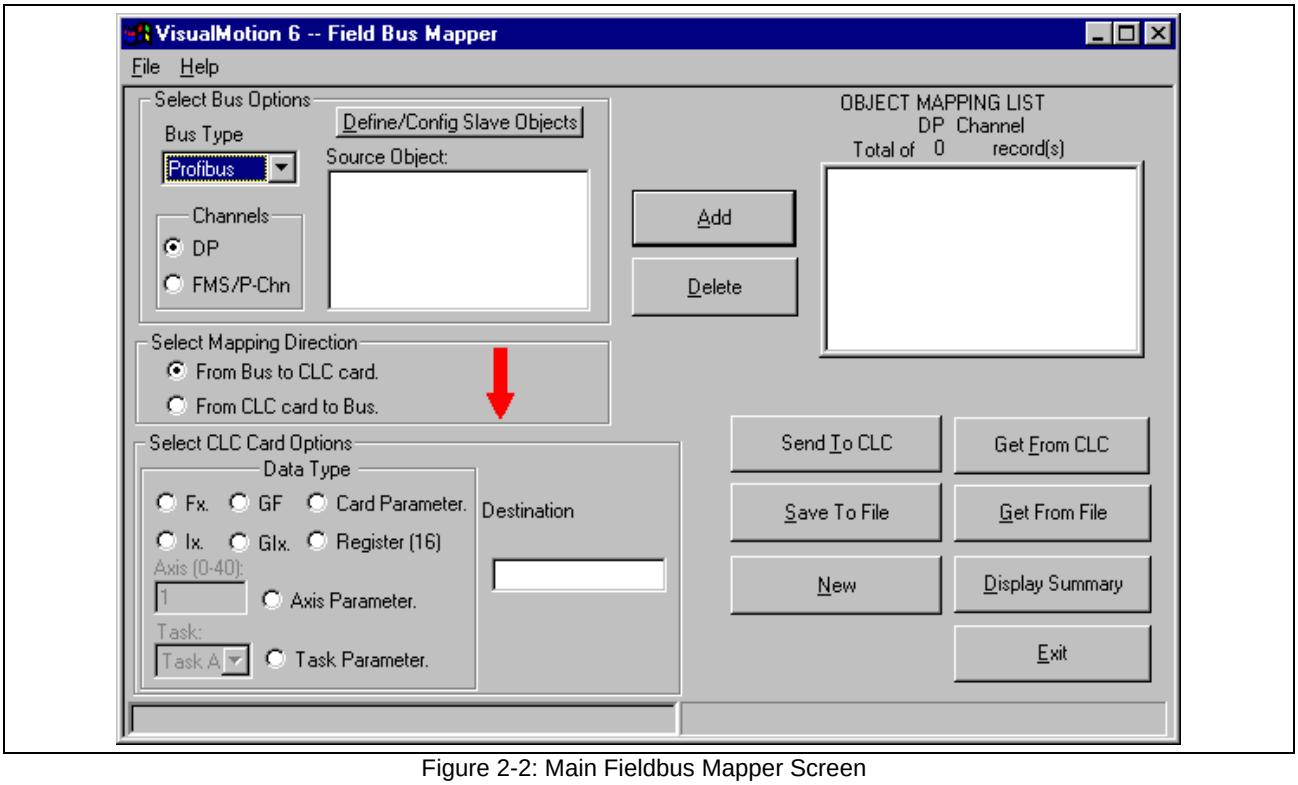


Figure 2-2: Main Fieldbus Mapper Screen

4. Click the button labeled "Define/Config Slave Objects." The Fieldbus Mapper will try to detect any configuration data from the CLC-D. If you are configuring a new list, click on the "NEW" button to clear any current data in the selected list. The Profibus Configuration Screen is pictured in ***Figure 2-3: Profibus Configuration Screen***.

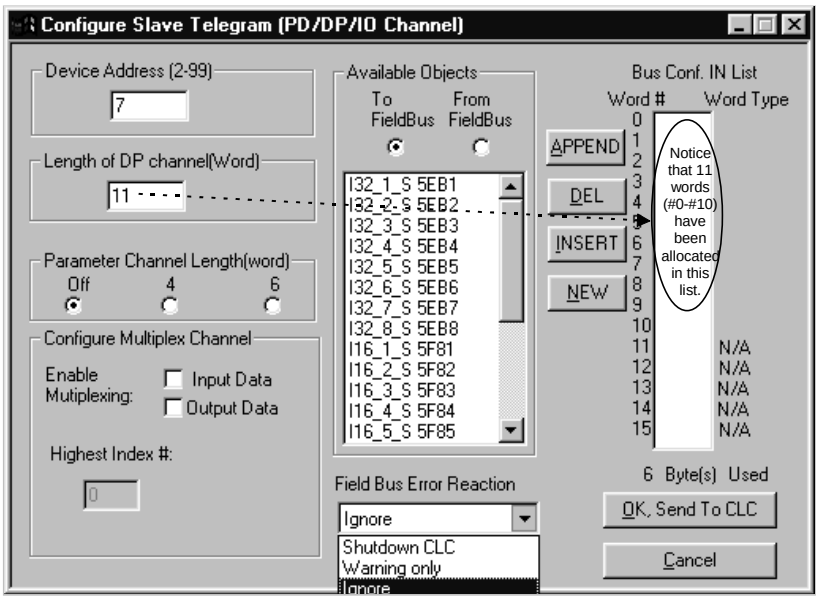


Figure 2-3: Profibus Configuration Screen

5. Set the device address to a unique number for this device on the bus (2-99).
6. Set the "Length of DP Channel (Word)" to the number of data words to be allocated for the input channel or the output channel, whichever is greater.
   According to the example data in ***Table 2-1: Sample Cyclic Data*** under "STEP I: Determine the Cyclic and Non-Cyclic Data," we have set the length of the channel to 11 for the output channel (made up of [3] 16-bit objects of 1 data word each and [4] 32-bit objects of 2 data words each). The input channel contains only 10 words, so a “filler” object must be placed in the Bus Configuration IN list for the 11th word.

Note: The information set in steps 5 and 6 is used in configuring the Fieldbus master (PLC) to look for the slave (CLC-D). Indramat supplies a *.gsd file containing supporting information for the CLC-D with DPF05.x card Profibus slave configuration. Contact an Indramat technical representative for location of this file.

7. Set the radio buttons under "Parameter Channel Length (word)" to "Off." If you want to use the parameter channel, see ***Parameter Channel Example***.
8. Ensure that both check boxes under "Configure Multiplex Channel" are unchecked. If you want to use multiplexing, see ***Multiplexing Example*** on page 2-11.
9. Ensure that the combo box under "Fieldbus Error Reaction" is currently set to "Ignore." Any other setting during setup will cause repeated error messages. Set the desired Error Reaction when the machine is ready for commissioning. See **Fieldbus Error Reaction** on page 3-9 for detailed information about the possible settings.
10. Choose the radio button below the words "To FieldBus."
11. For each data type desired, choose a corresponding available object in the list on the left and click "APPEND" to place it in the "Bus Conf. IN List." If you are configuring a new list, click on the "NEW" button to clear any current data in the selected list.
    Each available object has a typecode to identify its size and type.

Figure 2-4: Configuring the Profibus Configuration IN List
contains a description of the typecode.

Note: The desired order of the data on the master determines the object types (16- or 32-bit) and order of objects you place in the bus configuration list.

Note: Because the number of data objects in the IN list is less than the number in the OUT list, the remaining data word in the IN list must be assigned an available object that will not be used.

Following are descriptions of the buttons to manipulate the bus configuration lists:

inserts an available object after the last word placed in the current list.

removes the selected object from the current list.

inserts an available object above the selected object in the current list.

clears up the current list (only in the direction selected under “Available Objects”).

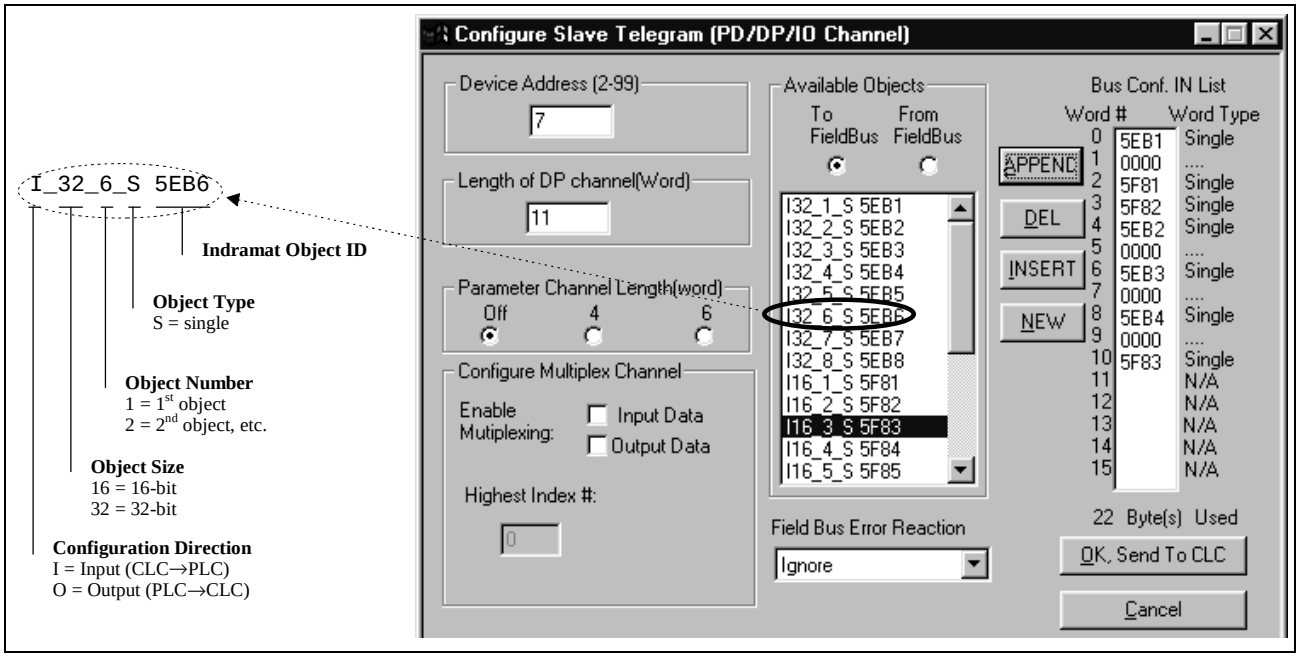


Figure 2-4: Configuring the Profibus Configuration IN List

12. Choose the radio button below the words "From FieldBus."

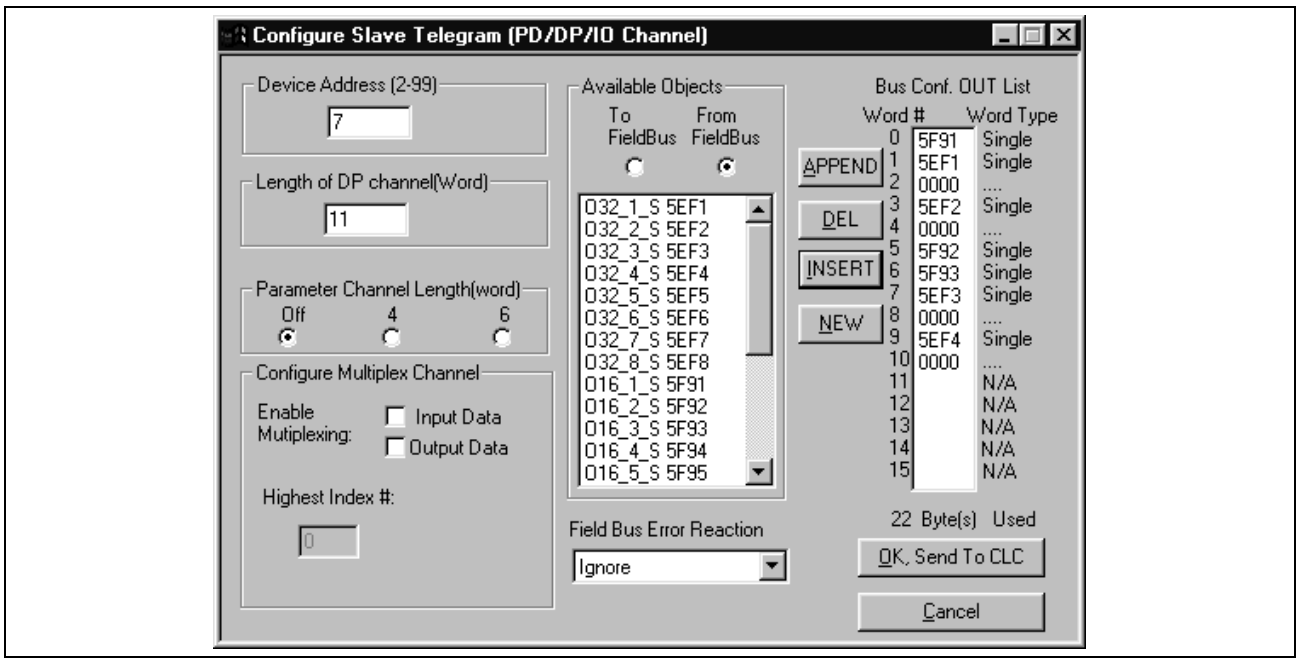


Figure 2-5: Configuring the Profibus Configuration OUT List

13. For each data type desired, choose a corresponding available object in the list on the left and click "APPEND" to place it in the "Bus Conf. OUT List." If you are configuring a new list, click on the "NEW" button to clear any current data in the selected list.

    We have now allocated the data objects in the cyclic channel so the desired data can be transferred between the CLC-D card and the Profibus Fieldbus card (see ***Figure 2-5: Configuring the Profibus Configuration OUT List***).

    Note: The non-cyclic data requires no configuration of objects in a list. These objects are addressed directly and preassigned as non-cyclic. They need only to be mapped to the CLC-D data.

14. Click "OK, Send to CLC." The following window appears:

Figure 2-6: Warning after clicking "OK, Send to CLC"

STEP III: Define Cyclic Data Mapping Lists

STEP III associates the CLC-D data types to the objects assigned in STEP II.

In our example, the objects to be added are:

Output Data (From Bus to CLC Card)	Assigned Object	Input Data (From CLC Card to Bus)	Assigned Object
Register 100	5F91	Integer 29	5EB1
Float 2	5EF1	Register 47	5F81
Global Integer 3	5EF2	Register 120	5F82
Register 104	5F92	Global Float 22	5EB2
Register 40	5F93	Axis Parameter 1.100	5EB3
Integer 4	5EF3	Integer 30	5F83
Integer 5	5EF4		

Adding Objects to the Cyclic Data Mapping List

1. Ensure that the designated bus type is correct (Profibus).
2. Choose the desired cyclic data channel (DP).
3. Select the desired Mapping Direction.
   Our example begins with "From Bus to CLC Card."
4. Select the Data Type.
   In our example, the first item to be added to this list is Register 100.
   Select "Register" as the data type.

Note: The CLC-D will show only the objects associated with the chosen data type in the Source Object list. For example, if you have selected the mapping direction "From Bus to CLC Card" and designate "Register" as the data type, only 16-bit output objects that have been mapped in the bus configuration list (see STEP II) will be shown in the Source Object list.

5. Select or fill in the Destination. In our example, we must select Register 100 from the list in the pop-up screen shown in ***Figure 2-7: Sample Selection List*** OR type 100 as the destination.

Note: The Fieldbus Mapper selection lists display only the labels of the currently active program.

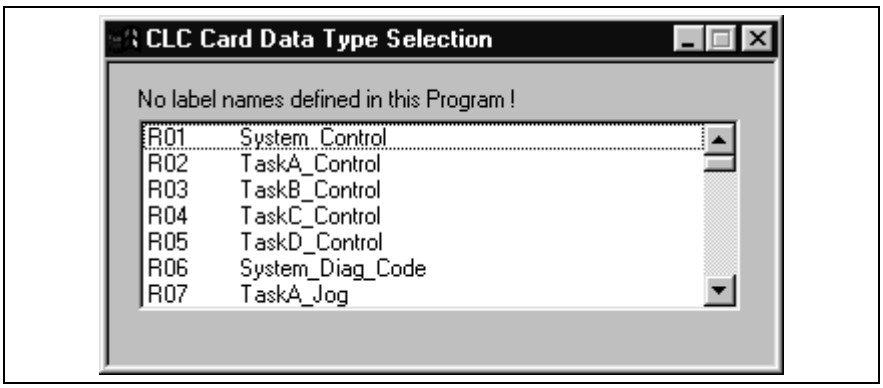


Figure 2-7: Sample Selection List

6. The source objects that are applicable to this data type will appear in the Source Object list. Choose the first available object.
7. Click the "Add" button. The object will appear in the "OBJECT MAPPING LIST" at the right.

Note: This step associates the object assigned in STEP II to the CLC-D data. The CLC-D scans the list created in this step and executes each mapping instruction **in order** during every third SERCOS update cycle.

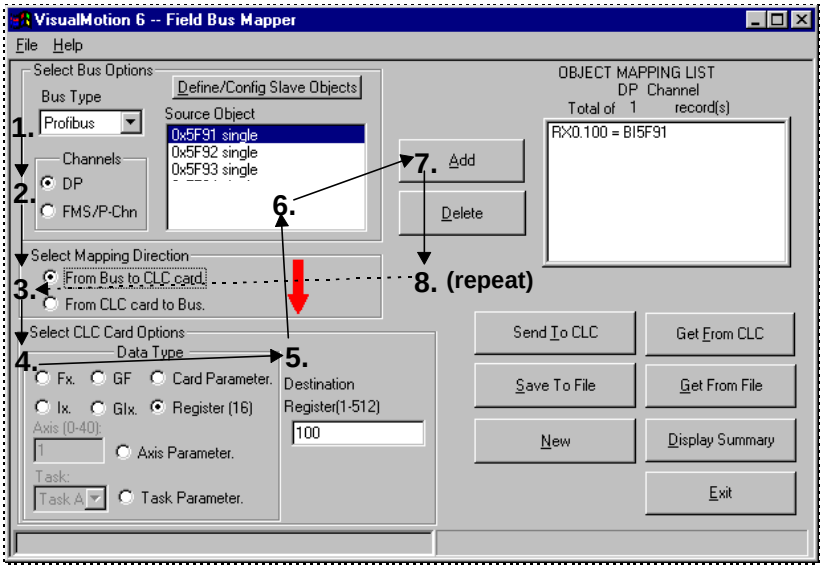


Figure 2-8: Adding Objects to the Cyclic Object Mapping List

Important: Cyclic data uses the objects configured in STEP II.

8. Repeat steps 3 through 7 above until each of the desired cyclic data objects has been added to the OBJECT MAPPING LIST.

Note: When mapping a PLC output object to a parameter, the parameter must be of the type "Read/Write Any Time." For parameters that are "Read Any Time/Write in Parameter Mode" or "Read Only," an error message is generated when you click “Send to CLC” if you try to map the parameters to a PLC output object. Examples: C-0-1550 (Read Any Time / Write in Parameter Mode) and C-0-1518 (Read Only).

9. Click “Send to CLC” to save the selected object mapping list to the CLC-D card. Use any of the following buttons, as needed.

Saves the currently selected object mapping list to the CLC-D Card (to card parameter C-0-2600 if the cyclic [DP] channel is selected, and C-0-2700 if the non-cyclic [FMS/P-Chan] channel is selected).

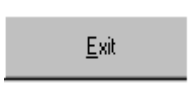
Saves the currently selected object mapping list to a file (with a .prm extension).

Clears up the current object mapping list.

Gets the currently selected object mapping list from the CLC-D Card (from card parameter C-0-2600 if the cyclic [DP] channel is selected, and C-0-2700 if the non-cyclic [FMS/P-Chan] channel is selected).

Allows the user to open a previously saved file (with a .prm extension) of an object mapping list. When the file is opened, displays that object mapping list.

Generates an on-line summary report of the currently selected Fieldbus. This report can be printed by selecting "Print" from the File menu in the Fieldbus Mapper window.



Exits the Fieldbus Mapper utility.

Important: Ensure that your system is set to Parameter Mode for the changes to take place.

Note: For debugging purposes, the Fieldbus Mapper follows each mapped item with the cursor as it is saved to the CLC-D. If there is a problem with the mapping of a particular object, an error message appears while the cursor remains on the object.

Changing an Existing Object Mapping List

If you want to insert, replace or delete items in the Object Mapping List, double-click on the item in question. A new set of buttons appears to the left of the Object Mapping List box.

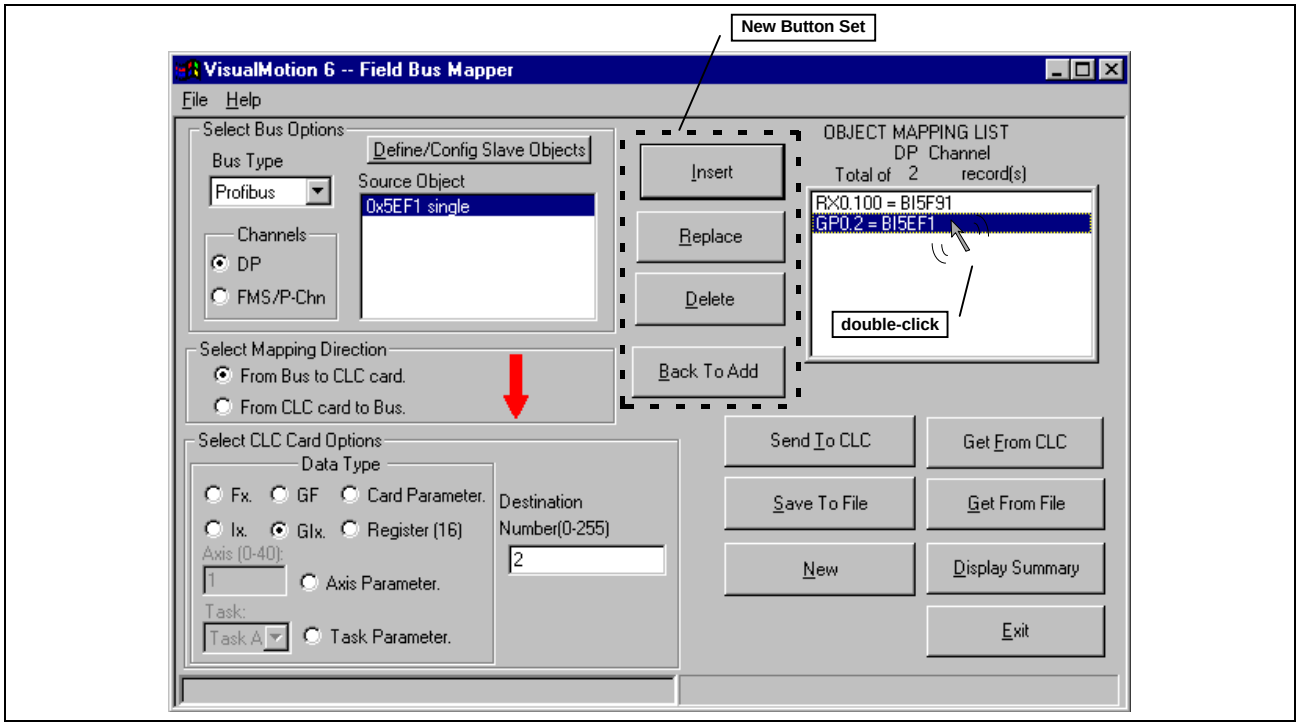


Figure 2-9: Changing an Existing Mapping List in the Fieldbus Mapper

Inserts a new object into the list directly before the selected object.

Replaces the selected object with a new object.

Removes the selected object from the list.

Returns to Fieldbus mapper normal mode, which allows adding new items to the end of the list and deleting items.

STEP IV: Define Non-Cyclic Data Mapping Lists (Direct Mapping)

The non-cyclic data uses objects that are pre-defined for this type of data.

In our example, the following data will be transmitted non-cyclically:

Output Data (From Bus to CLC Card)	Input Data (From CLC Card to Bus)
Float 16	Float 5
Integer 7	Integer 8
Register 101	

Table 2-3: Objects to be transferred non-cyclically

Adding Objects to the Non-Cyclic Data Mapping List (see *Figure 2-10: Adding Objects to the Non-Cyclic Object Mapping List*)

1. Ensure that the designated bus type is correct.
2. Choose the non-cyclic data channel (FMS/P-Chan).
3. Select the desired Mapping Direction.
   In our example, we will begin with the output list, or "From Bus to CLC."
4. Select the Data Type.
   In our example, the first item to be added to this list is Float 16. Select "Float" as the data type.
5. Fill in the Destination.
   In our example, we must type 16 as the destination.

6. The source objects that are applicable to this data type will appear in the Source Object list.
   Choose the first available object.
7. Click the "Add" button.
   The object will appear in the "OBJECT MAPPING LIST" at the right. This list is scanned by the CLC only when a non-cyclic request comes in and executes the mapping.
8. Repeat steps 2 through 7 above until each of the desired non-cyclic objects has been added to the OBJECT MAPPING LIST in both directions ("From Bus to CLC card" AND "From CLC card to Bus").

Note: If you want to insert, replace or delete items in the Object Mapping List, double-click on the item in question. A new set of buttons appears to the left of the Object Mapping List box. See ***Changing an Existing Object Mapping List*** on page 2-8 for a detailed explanation of each button.

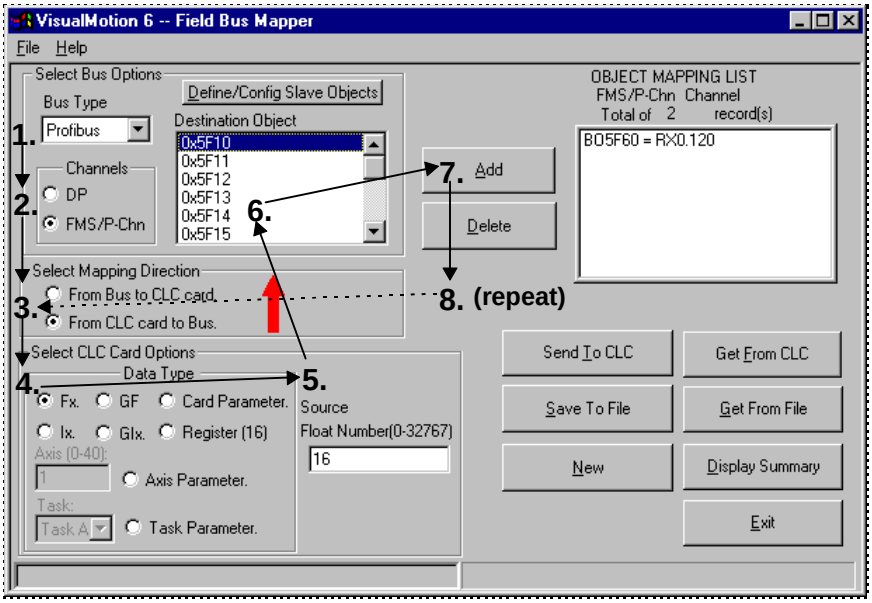


Figure 2-10: Adding Objects to the Non-Cyclic Object Mapping List

9. Click “Send to CLC” to save the selected object mapping list to the CLC-D card. Use any of the following buttons, as needed.

Send To CLC	Saves the currently selected object mapping list to the CLC-D Card (to card parameter C-0-2600 if the cyclic [DP] channel is selected, and C-0-2700 if the non-cyclic [FMS/P-Chan] channel is selected).
Save To File	Saves the currently selected object mapping list to a file (with a .prm extension).
New	Clears up the current object mapping list.
Get From CLC	Gets the currently selected object mapping list from the CLC-D Card (from card parameter C-0-2600 if the cyclic [DP] channel is selected, and C-0-2700 if the non-cyclic [FMS/P-Chan] channel is selected).
Get From File	Allows the user to open a previously saved file (with a .prm extension) of an object mapping list. When the file is opened, displays that object mapping list.
Display Summary	Generates an on-line summary report of the currently selected Fieldbus. This report can be printed by selecting "Print" from the File menu in the Fieldbus Mapper window.
Exit	Exits the Fieldbus Mapper utility.

Note: For debugging purposes, the Fieldbus Mapper follows each mapped item with the cursor as it is saved to the CLC-D. If there is a problem with the mapping of a particular object, an error message appears while the cursor remains on the object.

10. **Mapping is complete!**
    For programming information, refer to ***Information for the GPS Programmer*** on page 3-1or ***Information for the PLC Programmer*** on page 4-1.

2.2 Multiplexing Example

Multiplexing is available in the cyclic (DP) channel for applications where more than the allotted 16 data transfer words are required. For a more detailed description of multiplexing, see ***Multiplex Data Types*** (Cyclic Data Channel) on page 1-4.

Important: The Fieldbus Mapper is an on-line configuration tool. The following conditions must be met when performing STEP I through STEP IV below:
- The system must be powered up.
- The CLC-D card must be connected to the system.
- The system must be set to parameter mode.

STEP I: Determining the Cyclic and Non-Cyclic Data (with Multiplexing)

Cyclic Data

A typical multiplexing example in a multi-axis application is to assign cyclic data to each index by axis (e.g. index 0 to axis 1, index 1 to axis 2, etc.). Although this example shows data for only three axes in this manner, remember that multiplexing allows sending up to 16 unique pieces of data for each multiplex object. Single and multiplex objects can be combined to fill up the first 15 words of the list.

In this example, we will assume that you want to cyclically transfer the following data between the CLC-D card and the Fieldbus master in the order listed:

Object Type, Size (bits)	Output Data (From Fieldbus Master to CLC Card)			Size (Data Words)	Object Type, Size (bits)	Input Data (From CLC Card to Fieldbus Master)			Size (Data Words)
	Axis 1 (Index 0)	Axis 2 (Index 1)	Axis 3 (Index 2)			Axis 1 (Index 0)	Axis 2 (Index 1)	Axis 3 (Index 2)	
Single, 16-bit	Register 100			1	Single, 16-bit	Register 120			1
Single, 32-bit	Float 2			2	Single, 32-bit	Global Float 22			2
Single, 32-bit	Integer 4			2	Single, 32-bit	Global Integer 44			2
Multiplex, 16-bit	Control Registers (preassigned per axis)				Multiplex, 16-bit	Status Registers (preassigned per axis)			
	Register 11	Register 12	Register 13	1		Register 31	Register 32	Register 33	1
Multiplex, 32-bit	Position Command (Floats called out in VisualMotion program)*see Note				Multiplex, 32-bit	Axis Parameter 102 (Feedback Position)			
	Float 11	Float 21	Float 31	2		Parameter A-1.102	Parameter A-2.102	Parameter A-3.102	2
Multiplex, 32-bit	Counter (Integers called out in VisualMotion program)*see Note				Multiplex, 32-bit	Axis Parameter 112 (Feedback Velocity)			
	Integer 11	Integer 21	Integer 31	2		Parameter A-1.100	Parameter A-2.100	Parameter A-3.100	2
Multiplex, 32-bit	Dwell Time (Integers called out in VisualMotion program)*see Note				Multiplex, 32-bit	Maximum Attempts (Integers called out in VisualMotion program) *see Note			
	Integer 14	Integer 24	Integer 34	2		Integer 13	Integer 23	Integer 33	2
				TOTAL: 12 Data Words					**TOTAL:** 12 Data Words

Note: We suggest devising a system for assigning multiplex integers and floats to particular axes or data sets. (However, there is no limitation on how to divide multiplex objects, except size.)In our sample data, the system is as follows:
1st digit: Axis number
2nd digit: Purpose (e.g. position, counter, dwell)

For example: Integers in above sample data

3 4

1=Axis 1, 2=Axis 2, 3=Axis 3... 1=Counter, 4=Dwell Time

Table 2-4: Sample Cyclic Data (with Multiplexing)

Non-Cyclic Data

We will assume that the following data will be transferred non-cyclically for system initialization or setup. Remember, the direct-mapped non-cyclic data is intended for infrequent data transfer.

Note: See ***Table 1-3: Non-Cyclic Data Objects*** for data limitations.

Output Data (From Fieldbus Master to CLC Card)	Size (Data Words)	Size (Bits)	Input Data (From CLC Card to Fieldbus Master)	Size (Data Words)	Size (Bits)
Float 16	2	32	Float 5	2	32
Integer 7	2	32	Integer 8	2	32
Register 101	1	16			

Table 2-5: Sample Non-Cyclic Data (with Multiplexing)

Note: For other data types, such as diagnostic text messages, or S and P parameters, use the data exchange object.

STEP II: Configure Fieldbus Slave Card with Setup Information and Cyclic Data Object Lists Using the Fieldbus Mapper

Important: The drive must be set to Parameter Mode to configure and assign the data mapping for Fieldbuses (STEP II and STEP III).

1. In the VisualMotion "Data" Menu, select "Fieldbus Mapper."

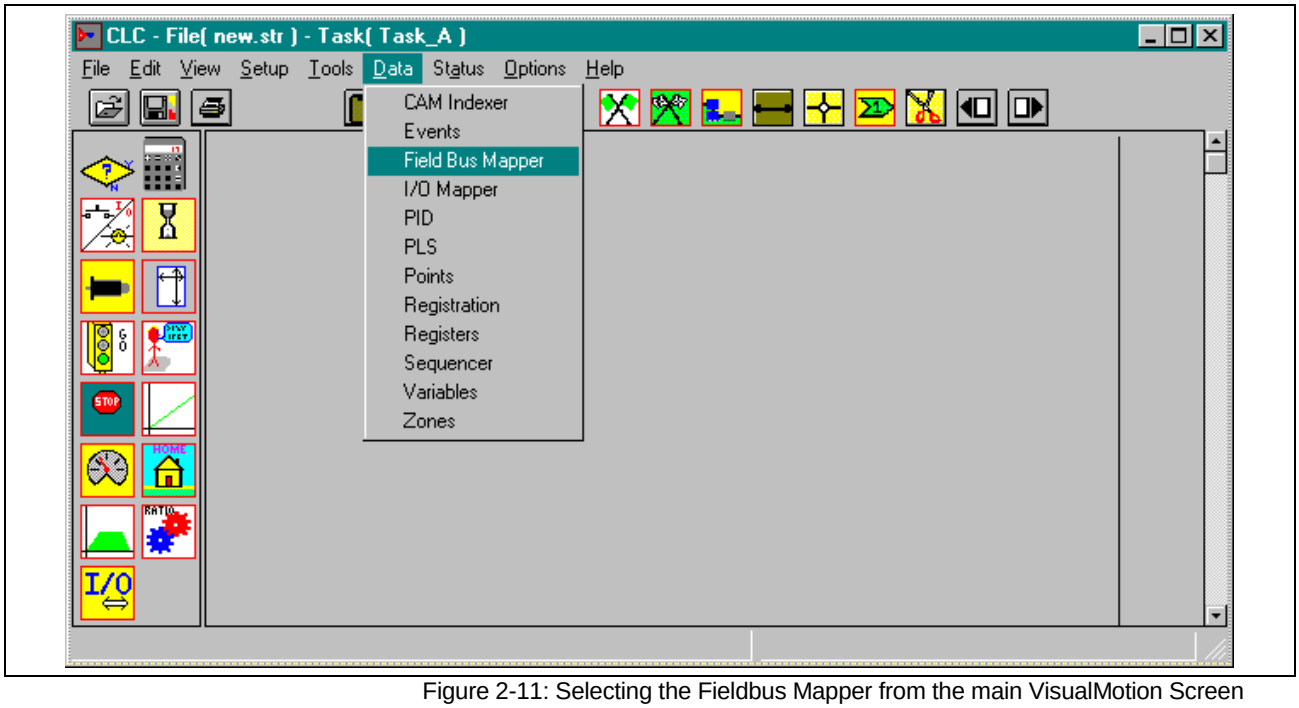


Figure 2-11: Selecting the Fieldbus Mapper from the main VisualMotion Screen

2. In the scroll box under "Bus Type," choose the desired bus (Profibus). The bus type may appear automatically if the Fieldbus card is connected and the firmware version is recognized.
3. Choose the cyclic data channel (DP).

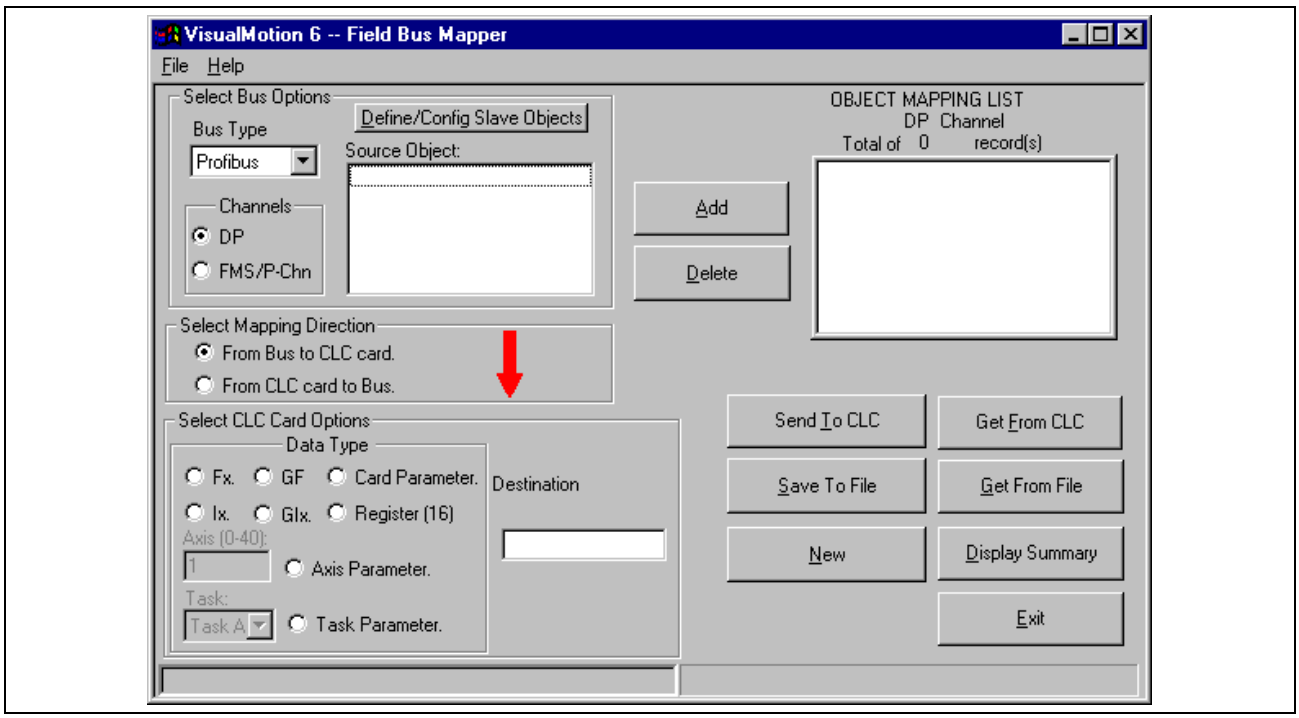


Figure 2-12: Main Fieldbus Mapper Screen

4. Click the button labeled "Define/Config Slave Objects." The Fieldbus Mapper will try to detect any configuration data from the CLC-D. If you are configuring a new list, click on the "NEW" button to clear any current data in the selected list. The Profibus Configuration Screen is pictured in ***Figure 2-13: Profibus Configuration Screen***.

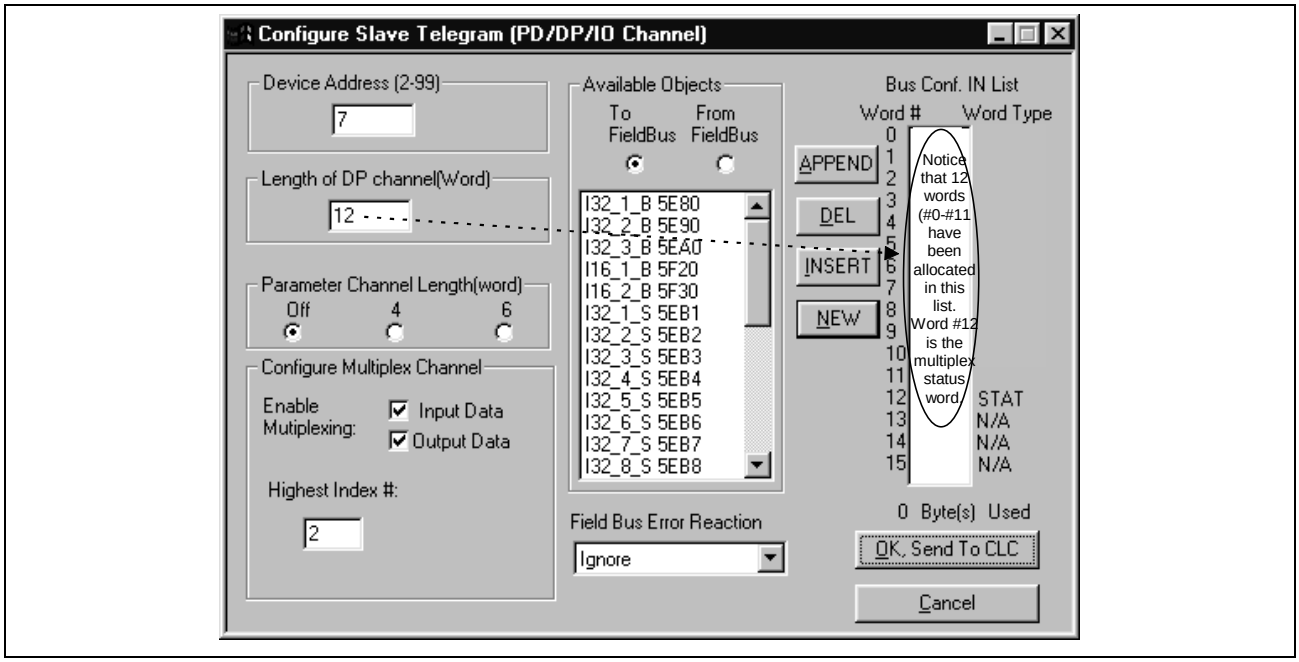


Figure 2-13: Profibus Configuration Screen

5. Set the device address to a unique number for the devices on the bus (2-99).
6. Set the "Length of DP Channel (Word)" to the number of data words to be allocated for the input channel or the output channel, whichever is greater. According to the example data in ***Table 2-4: Sample Cyclic Data (with Multiplexing)*** under "STEP I: Determine the Cyclic and Non-Cyclic Data," we have set the length of the channel to 12 for the output and input channels.

Note: The information set in steps 5 and 6 is used in configuring the Fieldbus master (PLC) to look for the slave (CLC-D). Indramat supplies a *.gsd file containing supporting information for the CLC-D with DPF05.x card Profibus slave configuration. Contact an Indramat technical representative for location of this file.

7. Set the radio buttons under "Parameter Channel Length (word)" to "Off." If you want to use the parameter channel, see ***Parameter Channel Example*** on page 2-20.
8. Click in both check boxes next to "Enable Multiplexing," because we want to configure multiplex objects in both the output and input channels.

   Notice that the 13th word (word #12) shows the status/control word (STAT in the Bus Conf. IN list, CNTL in the Bus Conf. OUT list). The bus now actually has 13 words in each of the bus configuration lists: the 12 words that were set on the left, plus the status/control word. See ***Multiplex Control and Status Words*** on page 1-5 for information about the usage of this extra word when accessing data from the Fieldbus master (PLC).
7. Fill in the highest index number as "2," because we are using only 3 sets of multiplex data (Indexes 0-2, for axes 1-3). The index number will always be one less than the number of indices, because the first index number is "0."
8. Ensure that the combo box under "Fieldbus Error Reaction" is currently set to "Ignore." Any other setting during setup will cause repeated error messages. Set the desired Error Reaction when the machine is ready for commissioning. See Fieldbus Error Reaction on page 3-9 for detailed information about the possible settings.
9. Choose the radio button below the words "To FieldBus."
10. Click “NEW” to clear up any existing data in the current list.

Note: The desired order of the data on the master determines the object types (16- or 32-bit) and order of objects you place in the bus configuration list. Therefore, care should be taken when placing these objects.

11. For each data type desired, choose a corresponding available object in the list on the left and click "APPEND" to place it at the end of the "Bus Conf. IN List."
    Each available object has a typecode to identify its size and type.
    Figure 2-14: Configuring the Profibus Configuration IN List contains a description of the typecode.

Following are descriptions of the buttons to manipulate the bus configuration lists:

inserts an available object after the last word placed in the current list.

removes the selected object from the current list.

inserts an available object above the selected object in the current list.

clears up the current list (only in the direction selected under “Available Objects”).

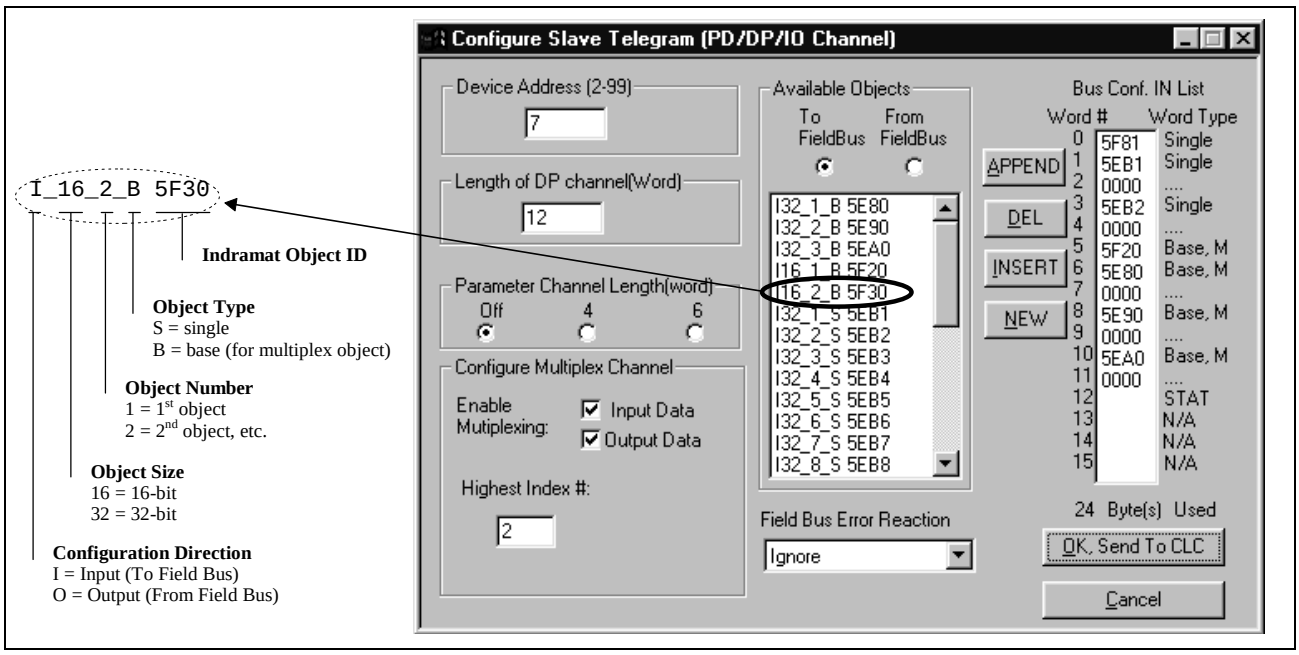


Figure 2-14: Configuring the Profibus Configuration IN List (with Multiplexing)

12. Choose the radio button below the words "From FieldBus."

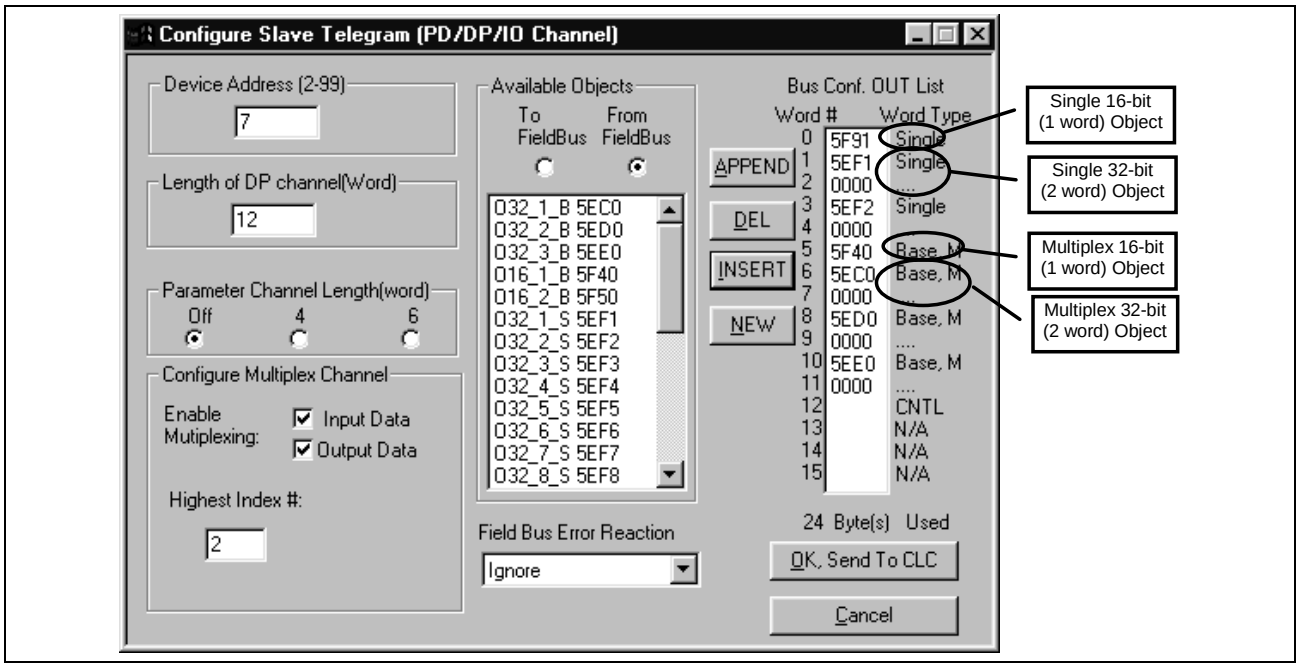


Figure 2-15: Configuring the Profibus Configuration OUT List (with Multiplexing)

13. For each data type desired, choose a corresponding available object in the list on the left and click "APPEND" to place it in the "Bus Conf. OUT List."

We have now allocated the data objects in the cyclic channel so the desired data can be transferred between the CLC-D card and the Profibus Fieldbus card.

Note: The non-cyclic data requires no configuration of objects in a list. These objects are addressed directly and preassigned as non-cyclic. They need only to be mapped to the CLC-D data.

14. Click "OK, Send to CLC." The following window appears:

Figure 2-16: Warning after clicking "OK, Send to CLC"

STEP III: Define Cyclic Data Mapping Lists

STEP III associates the CLC-D data types to the objects assigned in STEP II. In our example, the objects to be added are:

Object Type	Output Data (From Bus to CLC Card)			Input Data (From CLC Card to Bus)		
Single	Register 100			Register 120		
"	Float 2			Global Float 22		
"	Integer 4			Global Integer 44		
Multiplex	Register 11	Register 12	Register 13	Register 31	Register 32	Register 33
"	Float 11	Float 21	Float 31	Parameter A-1.102	Parameter A-2.102	Parameter A-3.102
"	Integer 11	Integer 21	Integer 31	Parameter A-1.110	Parameter A-2.110	Parameter A-3.110
"	Integer 14	Integer 24	Integer 34	Integer 13	Integer 23	Integer 33

Table 2-6: Objects to be transferred cyclically (with multiplexing)

Adding Objects to the Cyclic Data Mapping List

1. Ensure that the designated bus type is correct.
2. Choose the desired cyclic data channel (DP).
3. Select the desired Mapping Direction.
   Our example begins with "From Bus to CLC Card."
4. Select the Data Type.
   In our example, the first single item to be added to this list is Register 120. Select "Register" as the data type.

Note: The CLC-D will show only the objects associated with the chosen data type in the Source Object list. For example, if you have selected the mapping direction "From Bus to CLC Card" and designate "Register" as the data type, only 16-bit output objects that have been mapped in the bus configuration list (see STEP II) will be shown in the Source Object list.

5. Select or fill in the Destination.
   In our example, we must select Register 120 from the list in the pop-up screen shown in ***Figure 2-7: Sample Selection List***, OR type 120 as the destination.

Note: The Fieldbus Mapper selection lists display only the labels of the currently active program.

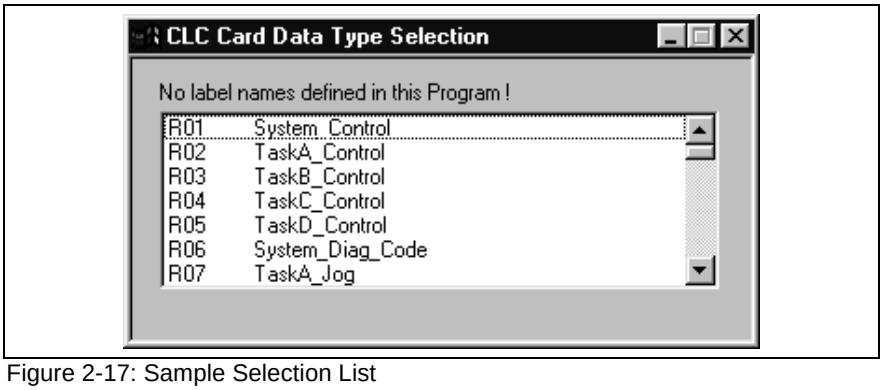


Figure 2-17: Sample Selection List

6. The source objects that are applicable to this data type will appear in the Source Object list.
   Choose the first available object.
7. Click the "Add" button.
   The object will appear in the "OBJECT MAPPING LIST" at the right.

Note: This step associates the object assigned in STEP II to the CLC-D data. The CLC-D scans the list created in this step and executes each mapping instruction **in order** every third SERCOS update cycle. Multiplex data can only be updated when this index is commanded by the multiplex control word.

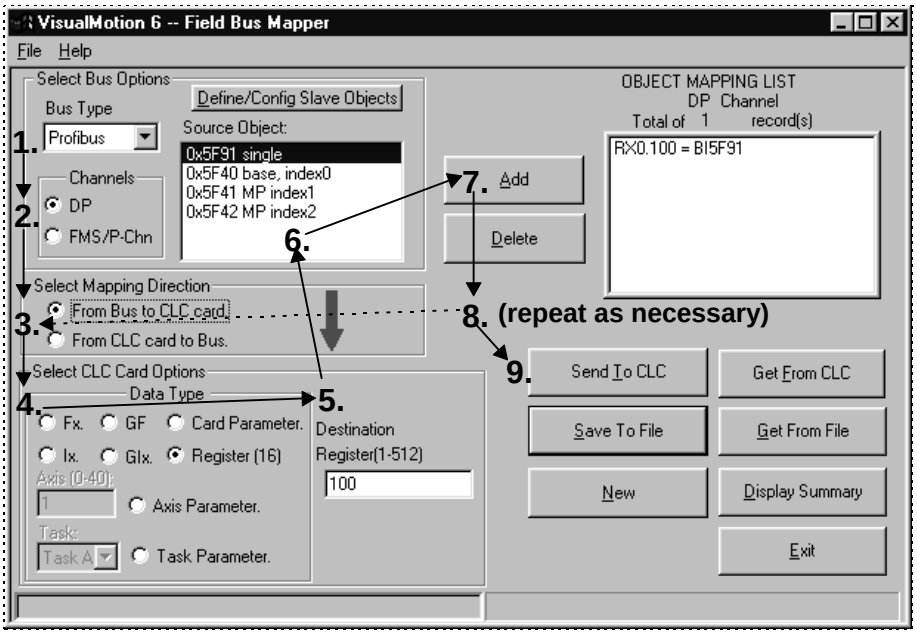


Figure 2-18: Adding Objects to the Cyclic Object Mapping List

Important: Cyclic data uses the objects configured in STEP II.

8. Repeat steps 3 through 7 above until each of the desired cyclic data objects has been added to the OBJECT MAPPING LIST.

Note: When mapping a PLC output object to a parameter, the parameter must be of the type "Read/Write Any Time." For parameters that are "Read Any Time/Write in Parameter Mode" or "Read Only," an error message is generated when you click “Send to CLC” if you try to map the parameters to a PLC output object. Examples: C-0-1550 (Read Any Time / Write in Parameter Mode) and C-0-1518 (Read Only).

Note: If you want to insert, replace or delete items in the Object Mapping List, double-click on the item in question. A new set of buttons appears to the left of the Object Mapping List box. See ***Changing an Existing Object Mapping List*** on page 2-8 for a detailed explanation of each button.

9. Click “Send to CLC” to save the selected object mapping list to the CLC-D card. Use any of the following buttons, as needed.

Saves the currently selected object mapping list to the CLC-D Card (to card parameter C-0-2600 if the cyclic [DP] channel is selected, and C-0-2700 if the non-cyclic [FMS/P-Chan] channel is selected).

Saves the currently selected object mapping list to a file (with a .prm extension).

Clears up the current object mapping list.

Gets the currently selected object mapping list from the CLC-D Card (from card parameter C-0-2600 if the cyclic [DP] channel is selected, and C-0-2700 if the non-cyclic [FMS/P-Chan] channel is selected).

Allows the user to open a previously saved file (with a .prm extension) of an object mapping list. When the file is opened, displays that object mapping list.

Generates an on-line summary report of the currently selected Fieldbus. This report can be printed by selecting "Print" from the File menu in the Fieldbus Mapper window.

Exits the Fieldbus Mapper utility.

Important: Ensure that your system is set to Parameter Mode for the changes to take place.

STEP IV: Define Non-Cyclic Data Mapping Lists (Direct Mapping)

The non-cyclic data uses objects that are pre-defined for this type of data.

In our example, the following data will be transmitted non-cyclically:

Output Data (From Bus to CLC Card)	Input Data (From CLC Card to Bus)
Float 16	Float 5
Integer 7	Integer 8
Register 101	

Table 2-7: Objects to be transferred non-cyclically

Adding Objects to the Non-Cyclic Data Mapping List (see *Figure 2-19: Adding Objects to the Non-Cyclic Object Mapping List*)

1. Ensure that the designated bus type is correct.
2. Choose the non-cyclic data channel (FMS/P-Chan).
3. Select the desired Mapping Direction.
   In our example, we will begin with the output list, or "From Bus to CLC."
4. Select the Data Type. In our example, the first item to be added to this list is Float 16. Select "Float" as the data type.
5. Fill in the Destination. In our example, we must type 16 as the destination.
6. The source objects that are applicable to this data type will appear in the Source Object list. Choose the first available object.
7. Click the "Add" button.
   The object will appear in the "OBJECT MAPPING LIST" at the right. This list is scanned by the CLC only when a non-cyclic request comes in and executes the mapping.
8. Repeat steps 2 through 7 above until each of the desired non-cyclic objects has been added to the OBJECT MAPPING LIST in both directions ("From Bus to CLC card" AND "From CLC card to Bus").

Note: If you want to insert, replace or delete items in the Object Mapping List, double-click on the item in question. A new set of buttons appears to the left of the Object Mapping List box. See ***Changing an Existing Object Mapping List*** on page 2-8 for a detailed explanation of each button.

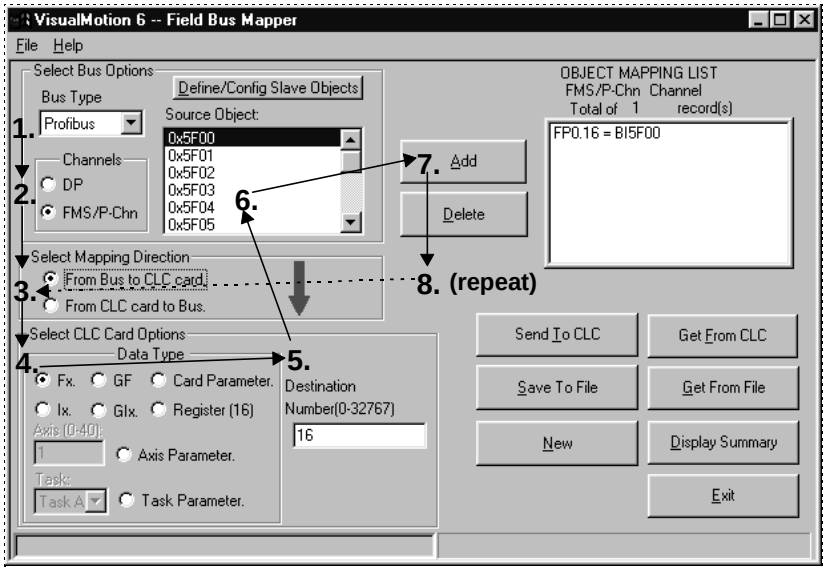


Figure 2-19: Adding Objects to the Non-Cyclic Object Mapping List

9. Click “Send to CLC” to save the selected object mapping list to the CLC-D card. Use any of the following buttons, as needed.

Saves the currently selected object mapping list to the CLC-D Card (to card parameter C-0-2600 if the cyclic [DP] channel is selected, and C-0-2700 if the non-cyclic [FMS/P-Chan] channel is selected).

Saves the currently selected object mapping list to a file (with a .prm extension).

Clears up the current object mapping list.

Gets the currently selected object mapping list from the CLC-D Card (from card parameter C-0-2600 if the cyclic [DP] channel is selected, and C-0-2700 if the non-cyclic [FMS/P-Chan] channel is selected).

Allows the user to open a previously saved file (with a .prm extension) of an object mapping list. When the file is opened, displays that object mapping list.

Generates an on-line summary report of the currently selected Fieldbus. This report can be printed by selecting "Print" from the File menu in the Fieldbus Mapper window.

Exits the Fieldbus Mapper utility.

Note: For debugging purposes, the Fieldbus Mapper follows each mapped item with the cursor as it is saved to the CLC-D. If there is a problem with the mapping of a particular object, an error message appears while the cursor remains on the object.

10. **Mapping is complete!**
For programming information, refer to ***Information for the GPS Programmer*** on page 3-1or ***Information for the PLC Programmer*** on page 4-1.

2.3 Parameter Channel Example

Where support of non-cyclic (FMS) communication is not available over the Fieldbus, a subset of the cyclic (DP) channel can be allocated for non-cyclic communications (e.g. parameterization and extended diagnostic information). This subset of the cyclic channel is called the **parameter channel**.

The parameter channel can be set up in the Fieldbus Mapper tool of the VisualMotion software when a CLC-D card is used. The first 4 or 6 data words of the DP channel can be allocated for the parameter channel.

Note: If you are using only Short Format 2 messaging format, only 4 words are needed by the system. If you are using VisualMotion ASCII Format or a combination of both messaging formats, 4 or 6 words can be used, depending on the requirements of the real-time information to be transmitted. For more information about these messaging formats, see ***Parameter Channel*** (Profibus DPF05.x with CLC-D only) on page 1-6.
Remember that the maximum number of words available for the cyclic (DP) channel is 16 (15 if multiplexing is used).

To set up the parameter channel in the Fieldbus Mapper:

1. In the "Configure Slave Telegram" screen in the VisualMotion Fieldbus Mapper (from the "Data" menu in main VisualMotion screen, choose "Fieldbus Mapper", then click the "Define/Config Slave Objects" button), choose one of the radio buttons below "Parameter Channel Length (word)."

Note: For the value entered in "Length of DP channel (Word)," you must take into consideration the addition of 4 or 6 data words for the parameter channel. The words allocated for the parameter channel will appear first in the list, and any other data will follow. **In this example, we had to increase the total DP Channel length to 15 words to accommodate the parameter channel.**

Note: If the length of DP Channel you entered at the left is not long enough to accommodate both the parameter channel **and** any existing data, the following error message will appear after clicking “OK, Send to CLC”: “The length of Param Channel Plus Length of Bus List should same as PD channel Length.”

In ***Figure 2-20: Parameter Channel Setup in the Fieldbus Mapper***, the parameter channel length is set to 4 words. Thus, the first four data words in the selected Data Configuration List at the right are allocated as type "Param."

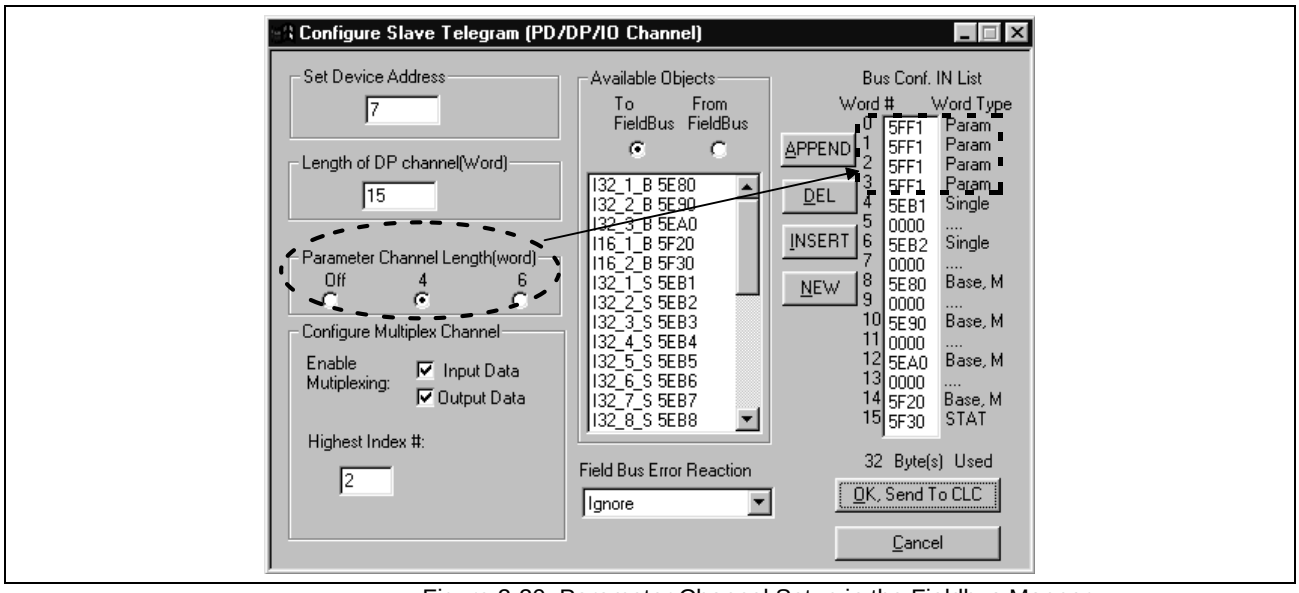


Figure 2-20: Parameter Channel Setup in the Fieldbus Mapper

2. Continue with the Fieldbus setup as indicated in one of the previous examples (***Basic Example*** beginning on page 2-1 or ***Multiplexing Example*** beginning on page 2-11).
   For detailed information about setup for the parameter channel-related data on the PLC side, see ***Information for the PLC Programmer*** on page 4-1.

3 Information for the GPS Programmer

3.1 Viewing/Printing a Summary Report of the Current Fieldbus (only when CLC-D is on-line)

To View a Summary Report:

From the Fielbus Mapper Window, click on the

button or select "Display Summary" from the File menu.

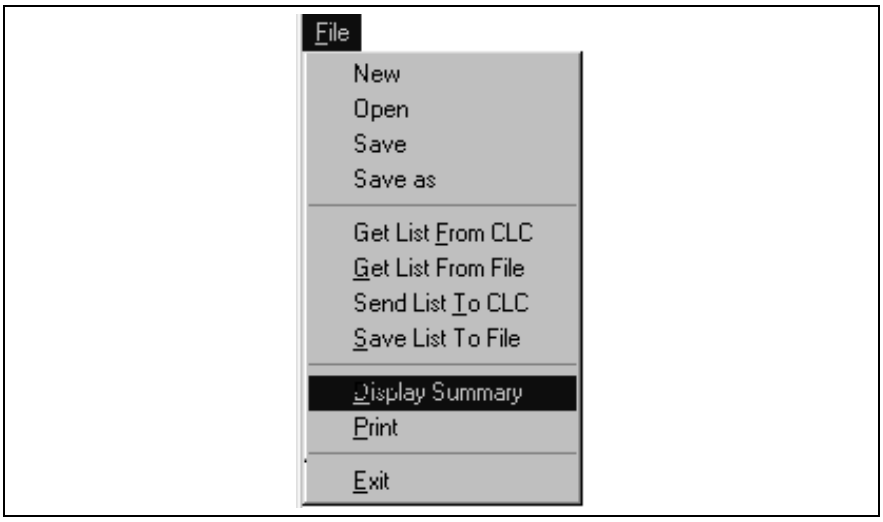


Figure 3-1: File Menu → Display Summary

A scrollable window will appear:

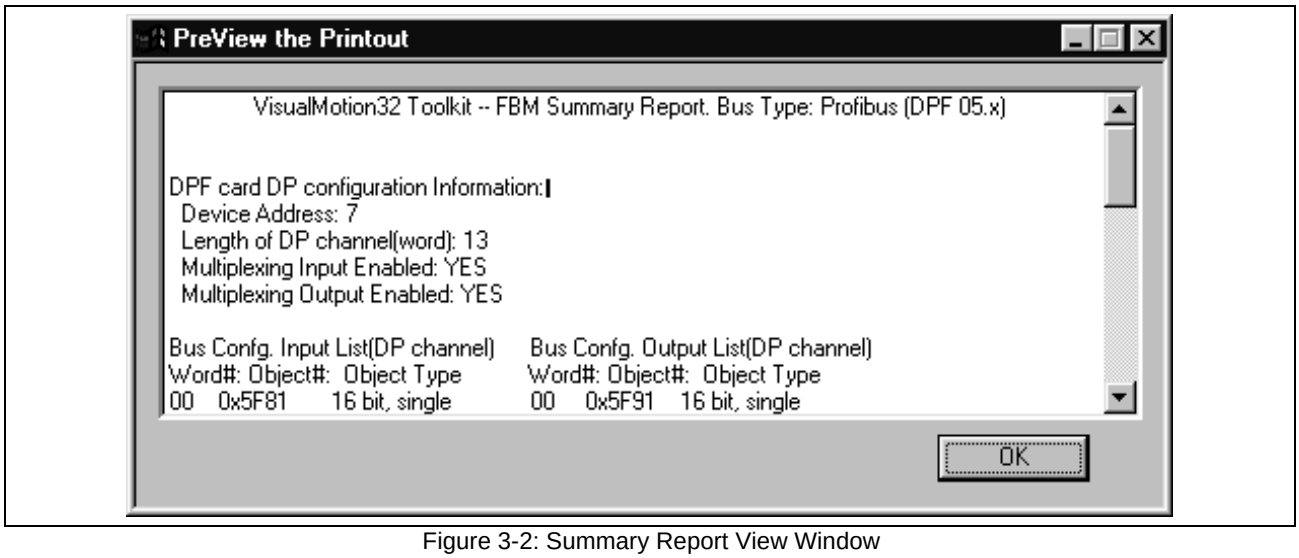

PreView the Printout
VisualMotion32 Toolkit -- FBM Summary Report. Bus Type: Profibus (DPF 05.x)
DPF card DP configuration Information:
Device Address: 7
Length of DP channel(word): 13
Multiplexing Input Enabled: YES
Multiplexing Output Enabled: YES
Bus Confg. Input List(DP channel) Bus Confg. Output List(DP channel)
Word#: Object#: Object Type Word#: Object#: Object Type
00 0x5F81 16 bit, single 00 0x5F91 16 bit, single
OK

Figure 3-2: Summary Report View Window

To Print a Summary Report:

From the Fieldbus Mapper window, select "Print" from the File menu.

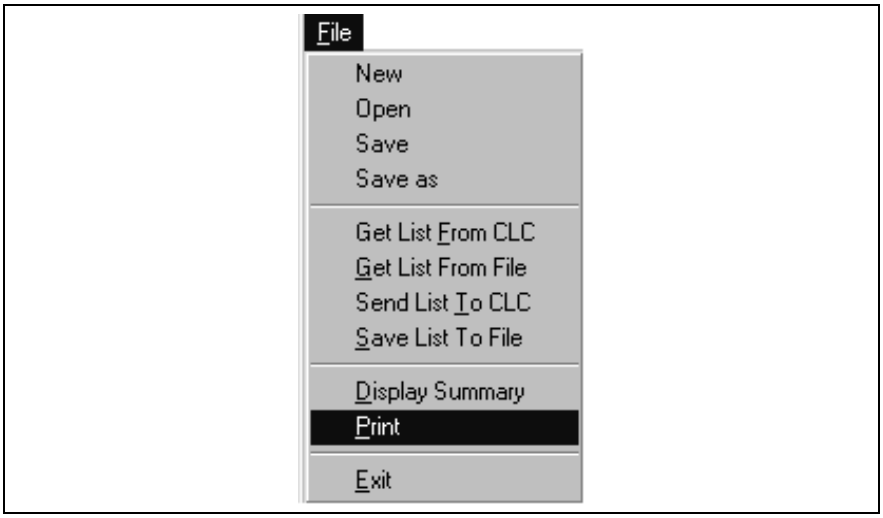


Figure 3-3: File Menu → Print

VisualMotion32 Toolkit – FBM Summary Report. Bus Type: Profibus (DPF 05.x)

Date: 05/05/99 Page 01 Time: 11:18:35

DPF card DP configuration Information:

Device Address: 7
Length of DP channel (word): 13
Multiplexing Input Enabled: YES
Multiplexing Output Enabled: YES
Highest Multiplexing Index #: 2

Bus Confg. Input List (DP channel)

Word#:	Object#:	Object Type
00	0x5F81	16 bit, single
01	0x5EB1	32 bit, single
02	0x0000	.....
03	0x5EB2	32 bit, single
04	0x0000	.....
05	0x5F20	16 bit, baseMP
06	0x5E80	32 bit, baseMP
07	0x0000	.....
08	0x5E90	32 bit, baseMP
09	0x0000	.....
10	0x5EA0	32 bit, base MP
11	0x0000	.....
12	0x5FFd	MP StatusWord

Bus Confg. Output List (DP channel)

Word#:	Object#:	Object Type
00	0x5F91	16 bit, single
01	0x5EF1	32 bit, single
02	0x0000	.....
03	0x5EF2	32 bit, single
04	0x0000	.....
05	0x5F40	16 bit, baseMP
06	0x5EC0	32 bit, baseMP
07	0x0000	.....
08	0x6ED0	32 bit, baseMP
09	0x0000	.....
10	0x5EE0	32 bit, baseMP
11	0x0000	.....
12	0x5FFC	MP ControlWord

CLC/GPS data mapping Lists for DPF card DP channel:

Data mapping IN List (DP channel)

Word#:	Data Type	Object Type
00	RX0.120	16 bit, single
01,02	HP0.22	32 bit, single
03,04	GP0.44	32 bit, single
05	RX0.31	16 bit, mult00
	RX0.32	16 bit, mult01
	RX0.33	16 bit, mult02
06,07	AP1.102	32 bit, mult00
	AP2.102	32 bit, mult01
	AP3.102	32 bit, mult02
08,09	AP1.100	32 bit, mult00
	AP2.100	32 bit, mult01
	AP3.100	32 bit, mult02
10,11	IP0.13	32 bit, mult00
	IP0.23	32 bit, mult01
	IP0.33	32 bit, mult02
12	0x5FFD	MP StatusWord

Data mapping OUT List (DP channel)

Word#:	Data Type	Object Type
00	RX0.100	16 bit, single
01,02	FP0.2	32 bit, single
03,04	IP0.4	32 bit, single
05	RX0.11	16 bit, mult00
	RX0.12	16 bit, mult01
	RX0.13	16 bit, mult02
06,07	FP0.11	32 bit, mult00
	FP0.21	32 bit, mult01
	FP0.31	32 bit, mult02
08,09	IP0.11	32 bit, mult00
	IP0.21	32 bit, mult01
	IP0.31	32 bit, mult02
10,11	IP0.14	32 bit, mult00
	IP0.24	32 bit, mult01
	IP0.34	32 bit, mult02
12	0x5FFC	MP ControlWord

CLC/GPS data mapping Lists for DPF card FMS channel:

Data mapping IN List(FMS channel)

Object#:	Data Type	Object Type
5F10	FP0.5	32 bit
5F11	IP0.8	32 bit

Data mapping OUT List(FMS channel)

Object#:	Data Type	Object Type
5F00	FP0.16	32 bit
5F01	IP0.7	32 bit
5F70	RX0.121	16 bit

Figure 3-4: Printout of Sample Fieldbus Mapping

3.2 Fieldbus-Accessible Parameters

Table 3-1 through Table 3-2 contain all of the Fieldbus-accessible parameters. Read and write access to these parameters is listed for both cyclic and non-cyclic data.

Parameter		Cyclic		Non-Cyclic (Direct-Mapped)	
Type/No.	Description	Read	Write	Read	Write
A-0-0020	Maximum Velocity	X (3)		X (3)	X (3)
A-0-0021	Maximum Acceleration	X		X	X
A-0-0022	Maximum Deceleration	X		X	X
A-0-0023	Jog Acceleration	X		X	X
A-0-0025	Maximum Jog Increment	X		X	X
A-0-0026	Maximum Jog Velocity	X		X	X
A-0-0031	CLC Cam/Ratio Master Factor (N)	X		X	
A-0-0032	CLC Cam/Ratio Slave Factor (M)	X		X	
A-0-0033	CLC Cam Stretch Factor (H)	X		X	
A-0-0034	CLC Cam Currently Active	X		X	X
A-0-0035	CLC Cam Position Constant (L)	X		X	
A-0-0037	Ratio Mode Step Rate	X		X	X
A-0-0038	Ratio Mode Options	X		X	
A-0-0100	Target Position	X (1, 3)		X (3)	
A-0-0101	Command Position	X (1, 3)		X (3)	
A-0-0102	Feedback Position	X (1, 3)		X (3)	
A-0-0110	Programmed Velocity	X (1, 3)	X (1, 3)	X (3)	X (3)
A-0-0111	Commanded Velocity	X (1, 3)		X (3)	
A-0-0112	Feedback Velocity	X (1, 3)		X (3)	
A-0-0120	Programmed Acceleration	X (1, 3)		X (3)	
A-0-0141	Torque Command	X (1, 3)		X (3)	
A-0-0142	Torque Feedback	X (1, 3)		X (3)	
A-0-0150	Programmed Ratio Adjust	X (1, 3)		X (3)	
A-0-0151	Programmed Phase Offset	X (3)		X (3)	
A-0-0153	CLC Phase Adjust Average Velocity	X (3)		X (3)	X (3)
A-0-0155	CLC Phase Adjust Time Constant	X		X	X
A-0-0157	Current Phase/CLC Cam Master Offset	X (3)		X (3)	
A-0-0159	Ratio Adjust Step Rate	X (3)		X (3)	X (3)
A-0-0160	Commanded Ratio Adjust	X (1, 3)		X (3)	
A-0-0164	ELS Options	X		X	
A-0-0180	Optional Command ID #1	X		X	
A-0-0181	Optional Command ID #2	X		X	
A-0-0182	Optional Command ID #3	X		X	
A-0-0185	Optional Feedback ID #1	X		X	
A-0-0186	Optional Feedback ID #2	X		X	
A-0-0190	Command Data #1	X (1, 2, 3)	X (1, 2, 3)	X (1, 2, 3)	X (1, 2, 3)
A-0-0191	Command Data #2	X (1, 2, 3)	X (1, 2, 3)	X (1, 2, 3)	X (1, 2, 3)
A-0-0192	Command Data #3	X (1, 2, 3)	X (1, 2, 3)	X (1, 2, 3)	X (1, 2, 3)
A-0-0195	Feedback Data #1	X (1, 2, 3)		X (1, 2, 3)	
A-0-0196	Feedback Data #2	X (1, 2, 3)		X (1, 2, 3)	
C-0-0001	Language Selection	X		X	
C-0-0002	Unit Number	X		X	X
C-0-0009	Error Reaction Mode	X		X	
C-0-0010	System Options	X		X	

1 = Must be configured by GPS in SERCOS telegram to update data. Otherwise, unexpected GPS system errors may occur.

2 = The data in this parameter is ONLY interpreted as type Float.

3 = This parameter is valid for fieldbus mapping in firmware version 06v64 and later. The data is read/written via the fieldbus in SERCOS phase 4 only. This parameter requires additional GPS resources to transfer data to/from the fieldbus. Therefore, care must be taken when using it for fieldbus communications with consideration to GPS system resources. Monitor GPS Register 26 for fieldbus mapping resource consumption, and increase the SERCOS update (C-0-0099) if necessary.

4 = No limit-checking is performed when writing to this parameter via a fieldbus. Therefore, care must be taken to ensure the proper data is sent.

Table 3-1: Fieldbus-Accessible Parameters (part 1 of 3)

Parameter		Cyclic		Non-Cyclic (Direct-Mapped)	
Type/No.	Description	Read	Write	Read	Write
C-0-0016	Communication Time-Out Period	X		X	X
C-0-0020	Transmitter Fiber Optic Length	X		X	X
C-0-0021	User Watchdog Timer	X		X	
C-0-0022	User Watchdog Task ID	X	X (4)	X	X
C-0-0030	Option Cards Active	X		X	
C-0-0031	Option Card Status	X		X	
C-0-0042	World Large Increment	X		X	X
C-0-0043	World Small Increment	X		X	X
C-0-0045	World Fast Jog Speed	X		X	X
C-0-0046	World Slow Jog Speed	X		X	X
C-0-0052	Axis Large Increment	X		X	X
C-0-0053	Axis Small Increment	X		X	X
C-0-0055	Axis Fast Jog Velocity	X		X	X
C-0-0056	Axis Slow Jog Velocity	X		X	X
C-0-0090	Download Block Size	X		X	X
C-0-0120	Operating Mode	X		X	
C-0-0121	SERCOS Communication Phase	X		X	
C-0-0123	Diagnostic Code	X		X	
C-0-0125	System Timer Value	X (3)		X	X
C-0-0151	Master 1 Drive Address	X		X	
C-0-0153	Virtual Master Programmed Velocity	X (3)		X (3)	X (3)
C-0-0154	Virtual Master Programmed Acceleration	X (3)		X (3)	X (3)
C-0-0155	Virtual Master Programmed Deceleration	X (3)		X (3)	X (3)
C-0-0156	Virtual Master E-Stop Deceleration	X (3)		X (3)	X (3)
C-0-0157	ELS Master Current Position	X (3)		X (3)	X (3)
C-0-0158	ELS Master Current Velocity	X (3)		X (3)	
C-0-0159	Master 1 Zero Velocity Window	X (3)		X (3)	X (3)
C-0-0160	Virtual Master Max. Jog Velocity	X		X	X
C-0-0161	Master 1 Ratio Input	X		X	
C-0-0162	Master 1 Ratio Output	X		X	
C-0-0164	Master 1 Encoder Type	X		X	
C-0-0170	PLS 1 Mask Register	X		X	X
C-0-0801	Pendant Protection Level 1 Password	X		X	X
C-0-0802	Pendant Protection Level 2 Password	X		X	X
C-0-0803	Pendant User Accessible Floats Section	X		X	X
C-0-0804	Pendant User Accessible Integers Section	X		X	X
C-0-0805	Pendant Start of User Accessible Registers	X		X	X
C-0-0806	Pendant End of User Accessible Registers	X		X	X
C-0-0807	Pendant Password Timeout	X		X	X
C-0-0810	TPT Message and Prompt Control Word	X	X (4)	X	X
C-0-0811	User Task Controlled Menu ID for TPT	X	X (4)	X	X
C-0-0812	User Task Controlled Task ID for TPT	X	X (4)	X	X
C-0-0813	User Task Controlled Axis Number for TPT	X	X (4)	X	X
C-0-0814	TPT Data Transaction Word	X	X (4)	X	X
C-0-1001	ELS Secondary Master	X		X	
C-0-1010	Master Switching Threshold	X (3)		X (3)	X (3)

1 = Must be configured by GPS in SERCOS telegram to update data. Otherwise, unexpected GPS system errors may occur.

2 = The data in this parameter is ONLY interpreted as type Float.

3 = This parameter is valid for fieldbus mapping in firmware version 06v64 and later. The data is read/written via the fieldbus in SERCOS phase 4 only. This parameter requires additional GPS resources to transfer data to/from the fieldbus. Therefore, care must be taken when using it for fieldbus communications with consideration to GPS system resources. Monitor GPS Register 26 for fieldbus mapping resource consumption, and increase the SERCOS update (C-0-0099) if necessary.

4 = No limit-checking is performed when writing to this parameter via a fieldbus. Therefore, care must be taken to ensure the proper data is sent.

Table 3-2: Fieldbus-Accessible Parameters (part 2 of 3)

Parameter		Cyclic		Non-Cyclic (Direct-Mapped)	
Type/No.	Description	Read	Write	Read	Write
C-0-1011	Diff. Between Real Master Positions	X (3)		X (3)	
C-0-1012	Master Synchronization Acceleration	X		X	
C-0-1013	Master Synchronization Time Constant	X		X	
C-0-1014	Master Synchronization Velocity Window	X		X	
C-0-1015	Virtual Master Current Position	X (3)		X (3)	X (3)
C-0-1016	Virtual Master Current Velocity	X (3)		X (3)	
C-0-1017	Position Difference Selection	X		X	X
C-0-1018	Position Monitoring Window	X (3)		X (3)	X (3)
C-0-1019	Position Difference	X (3)		X (3)	
C-0-1508	Master 1 Filter Cutoff Frequency	X		X	
C-0-1509	Master 1 Filter Type	X		X	
C-0-1517	Master 1 Current Position	X (3)		X (3)	X (3)
C-0-1518	Master 1 Current Velocity	X (3)		X (3)	
C-0-1550	Master 2 Type	X		X	
C-0-1551	Master 2 Drive Address	X		X	
C-0-1552	Master 2 Encoder Type	X		X	
C-0-1554	Master 2 Ratio Input	X		X	
C-0-1555	Master 2 Ratio Output	X		X	
C-0-1556	Master 2 Zero Velocity Window	X (3)		X (3)	X (3)
C-0-1558	Master 2 Filter Cutoff Frequency	X		X	
C-0-1559	Master 2 Filter Type	X		X	
C-0-1567	Master 2 Current Position	X (3)		X (3)	X (3)
C-0-1568	Master 2 Current Velocity	X (3)		X (3)	
C-0-2014	Start of SERCOS I-O Station Registers	X		X	
C-0-2015	Registers Allocated per I-O Station	X		X	
T-0-0002	Task Options	X		X	
T-0-0005	World Position Units	X		X	
T-0-0010	Kinematic Number	X		X	
T-0-0011	Coordinated X-Axis	X		X	
T-0-0012	Coordinated Y-Axis	X		X	
T-0-0013	Coordinated Z-Axis	X		X	
T-0-0020	Maximum Path Speed	X (3)		X (3)	
T-0-0021	Maximum Acceleration	X		X	
T-0-0022	Maximum Deceleration	X		X	
T-0-0023	Look Ahead Distance	X		X	X
T-0-0024	Velocity Override	X		X	X
T-0-0025	Maximum Jog Increment	X		X	X
T-0-0026	Maximum Jog Velocity	X		X	X
T-0-0035	Relative Point Used for Origin	X		X	X
T-0-0036	Relative Point Used for Tool Frame	X		X	X
T-0-0100	Target Point Number	X (3)		X (3)	
T-0-0101	Segment Status	X (3)		X (3)	
T-0-0102	Rate Limit Status	X (3)		X (3)	

1 = Must be configured by GPS in SERCOS telegram to update data. Otherwise, unexpected GPS system errors may occur.

2 = The data in this parameter is ONLY interpreted as type Float.

3 = This parameter is valid for fieldbus mapping in firmware version 06v64 and later. The data is read/written via the fieldbus in SERCOS phase 4 only. This parameter requires additional GPS resources to transfer data to/from the fieldbus. Therefore, care must be taken when using it for fieldbus communications with consideration to GPS system resources. Monitor GPS Register 26 for fieldbus mapping resource consumption, and increase the SERCOS update (C-0-0099) if necessary.

4 = No limit-checking is performed when writing to this parameter via a fieldbus. Therefore, care must be taken to ensure the proper data is sent.

Table 3-3: Fieldbus-Accessible Parameters (part 3 of 3)

3.3 Register 19 Definition (Fieldbus Status)

Diagnostic Object 5FF2

The diagnostic object 5ff2 holds the information for "Fieldbus Status," and this information is stored in VisualMotion Register 19. The register information can be referenced in a VisualMotion application program to respond to the status of each bit. The use of these bits is application-dependent.

Table 3-4: Bit Assignment for the Diagnostic Object 5ff2 (VisualMotion Register 19) contains the bit assignment for the diagnostic object 5ff2. The assigned bits are labeled with "x" and the bit number in the second row. Unassigned bits are labeled with "---."

16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
---	x15	---	x13	---	---	---	---	---	---	x6	x5	x4	---	x2	x1

Table 3-4: Bit Assignment for the Diagnostic Object 5ff2 (VisualMotion Register 19)

Bit Definitions

x1, x2 Status bits for the internal DPR (Dual-Port RAM) communication between the Fieldbus slave and the CLC-D:

x1: FB Init OK , LSB (least significant bit)

x2: FB Init OK, MSB (most significant bit)

The bit combinations for x1 and x2 are as follows:

Bit 2 (CLC-D)	Bit 1 (Fieldbus)	Description
0	0	A reset has been executed on the DPR, **or** neither the CLC-D card nor the Fieldbus card have initialized the DPR.
0	1	The DPR is initialized by the Fieldbus card, but **not** yet by the CLC-D card.
1	0	The DPR initialization is complete. DPR has been initialized by the Fieldbus card and CLC-D card.
1	1	Fieldbus to CLC-D communications system is ready. The system looks for this combination of bit settings to initiate any communication between the Fieldbus and CLC-D cards. The watchdog-function via DPR communications with the Fieldbus slave works correctly.

Table 3-5: Possible Settings for Bits 1 and 2, Status Bits for DPR Communication

x4 Status bit for the active bus capabilities of the Fieldbus slaves (FB Slave Ready)

This bit is monitored for the Fieldbus Error Reaction. Whenever this bit goes to 0 after a Fieldbus card was initially found by the CLC-D, the selected Error Reaction (system shutdown, error message, or ignore) is initiated. See **Fieldbus Error Reaction** on page 3-9 for an explanation of the Fieldbus Error Reaction setting.

0--> The Fieldbus slave is not (yet) ready for data exchange.

1--> The Fieldbus slave can actively participate on the bus.

x5 Status bit for the non-cyclic channel (FMS) (Non-Cyc Ready)

0--> The non-cyclic (FMS) channel can not (yet) be used.

1--> The non-cyclic (FMS) channel is ready for use by the Fieldbus Master.

x6 Status-bit for the reconfiguration of the cyclic (PD) channel (nCyc Chan Ready):

0--> The cyclic (PD) channel on the Fieldbus-card is configured properly. The system looks for this bit to be 0 before allowing data transfer.

1--> The cyclic (PD) channel is being reconfigured on the Fieldbus Card at this time.

x13 Status bit for non-supported GPS firmware (nVM FW OK):

0--> The GPS firmware version is supported with this Fieldbus interface.

1--> The GPS-firmware version is NOT supported by this Fieldbus interface card. The CLC-D hardware could support the Fieldbus interface, but the firmware version is not recognized as valid by the Fieldbus card firmware.

x15 Status bit for the cyclic data output (Cyclic Data Valid):

0--> The cyclic data outputs (coming in to the CLC-D) are INVALID.

1--> The cyclic data outputs (coming in to the CLC-D) are VALID. The system looks for this bit to be 1 before allowing data transfer.

3.4 Register 20 Definition (Fieldbus Diagnostics)

Diagnostic Object 5FF0

The diagnostic object 5ff0 holds the information for "Fieldbus Diagnostics," and this information is stored in VisualMotion Register 20.

Table 3-6: Bit Assignment for the Diagnostic Object 5ff0 (VisualMotion Register 20) contains the bit assignment for the diagnostic object 5ff0. The assigned bits are labeled with "x" and the bit number in the second row. Unassigned bits are labeled with "---."

16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
---	x15	x14	x13	---	---	---	---	x8	x7	x6	x5	x4	x3	x2	x1

Table 3-6: Bit Assignment for the Diagnostic Object 5ff0 (VisualMotion Register 20)

Bit Definitions

x1 - x8 Fieldbus Interface-specific fault code (FB Card Fault Code)

Note: This function is currently not available.

x13 - x15 Identification of the Fieldbus interface card (FB Card Found)

The bit combinations for x13, x14 and x15 are as follows:

Bit 15	Bit 14	Bit 13	Fieldbus Type
0	0	0	<NO CARD>
0	0	1	<Not Defined>
0	1	0	Interbus (DBS03)
0	1	1	DeviceNet (DCF01)
1	0	0	Profibus (DPF05)
1	0	1	CanOpen (DCF01)
1	1	0	<Not Defined>
1	1	1	<Not Defined>

Table 3-7: Identification of the Fieldbus Interface

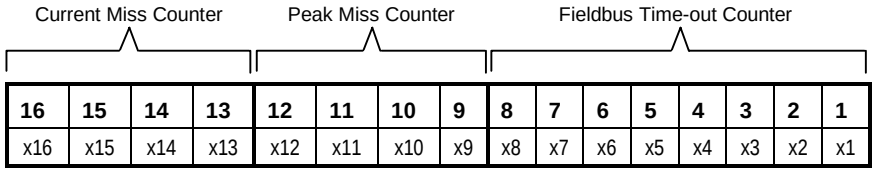
3.5 Register 26 Definition (Fieldbus Resource Monitor)

The "Fieldbus Resource Monitor" in register 26 can be used as a method for monitoring the attempts made to process the Object Mapping List parameter C-0-2600 across the Dual-port RAM. If after three SERCOS update cycles, the Object Mapping List is not successfully transmitted, a "**miss**" is noted.

Register 26 is divided into the following three counter types:

- Current Miss Counter.
- Peak Miss Counter.
- Fieldbus Timeout Counter.

Table 3-8: Bit Assignment for Fieldbus Resource Monitor (Register 26) contains the bit assignment for the Fieldbus counters. The assigned bits are labeled with an "x" followed by the bit number.



16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
x16	x15	x14	x13	x12	x11	x10	x9	x8	x7	x6	x5	x4	x3	x2	x1

Table 3-8: Bit Assignment for Fieldbus Resource Monitor (Register 26)

Note: View register 26 in Hexadecimal format to more easily monitor the Fieldbus counters.

Hex display format of register 26

0x0000

Fieldbus Time-out Counter (00-FF)
Increments by one every time 10 cyclic mapping list attempts are missed in 10 attempts.

Peak Miss Counter (0-9)
Displays the peak value of the Current Miss Counter.

Current Miss Counter (0-9)
Displays the count of missed cyclic mapping list attempts out of the last 10 attempts

Note: To view registers in hex, select ***Data ⇒ Registers*** within VisualMotion Toolkit. If the registers are not currently viewed in hex format, select ***Format ⇒ Hex*** in the *Active Program, Register* window.

Bit Definitions

x13 - x16 Status bits for the "Current Miss Counter."

An attempt is made to transmit the cyclic Object Mapping List, C-0-2600, across the Dual-port RAM every third SERCOS update cycle. For every 10 mapping list update attempts (30 SERCOS update cycles), the failed attempts are counted and displayed in these bits (values can range from 0-9). If 10 out of 10 mapping list updates attempts are missed, the "Fieldbus Timeout Counter" is incremented by one. This is an indication of a Fieldbus Mapping Timeout Error.

x9 - x12 Status bits for the "Peak Miss Counter."

These bits monitor the "Current Miss Counter's" peak count between a value from 0-9 and holds that value until a larger count is encountered.

x1 - x8 Status bits for the "Fieldbus Timeout Counter."

The count of these bits increments by one every time the "Current Miss Counter" encounters 10 out 10 missed attempts of the cyclic mapping list update. A count incremented by one represents a Fieldbus Mapping Timeout Error and is process by GPS according to the selected "Fieldbus Error Reaction" in parameter C-0-2635. See **Fieldbus Error Reaction** for an explanation of the Fieldbus Error Reaction setting.

Note: The GPS programmer can monitor the "Current Miss Counter" and define a custom error reaction for missed mapping list update attempts less than 10.

Note: The values in register 26 are read/write and can be reset by the user.

3.6 Fieldbus Error Reaction

Note: The Fieldbus Error Reaction setting is active only in SERCOS Phase 4. In all other SERCOS phases, it will be inactive.

You can choose how you would like the CLC-D system to react in case of a Fieldbus error. This reaction can be set in the "Configure Slave Telegram" screen, using the combo box labeled "Fieldbus Error Reaction."

Three options are available for the Error Reaction setting. Depending on the selected setting, the value 0, 1, or 2 is stored in Parameter C-0-2635:

Setting	Value in Parameter C-0-2635
Shutdown CLC	0 (default)
Warning Only	1
Ignore	2

Table 3-9: Parameter C-0-2635 Values for Error Reaction Settings

Fieldbus Mapper Timeout The Fieldbus Mapper continually scans the system for sufficient resources to process the cyclic data mapping list (2600-list). If 10 out of 10 attempts of the mapping list update are missed, the system is

considered to have insufficient resources and the selected error reaction is evoked, as follows:

If "Shutdown CLC" (0) is set in Parameter C-0-2635, the following error is generated from the CLC-D card: **520 Fieldbus Mapper Timeout**

If "Warning Only" (1) is set in Parameter C-0-2635, the following error is generated: **209 Fieldbus Mapper Timeout**

If "Ignore" (2) is set in Parameter C-0-2635, the system will update as resources become available, but there is no way to monitor whether or not updates actually occur.

Lost Fieldbus Connection

Object 5ff2, the dual-port RAM location on the Fieldbus card, indicates the status of the Fieldbus. Register 19 Bit 4, the Fieldbus Slave Ready Bit on the CLC card, mirrors object 5ff2. See ***3.3 Register 19 Definition (Fieldbus Status)*** for more specific bit information. The system monitors this bit and evokes the selected error reaction if the bit is low (0), after a Fieldbus card is found. A typical situation that will cause this condition is the disconnection of the Fieldbus cable from the DPF card.

If "Shutdown CLC" (0) is set in Parameter C-0-2635, the following error is generated from the CLC-D card (active in SERCOS Phase 4 only): **519 Lost Fieldbus Connection**

If "Warning Only" (1) is set in Parameter C-0-2635, the following error is generated (active in SERCOS Phase 4 only): **208 Lost Fieldbus Connection**

If "Ignore" (2) is set in Parameter C-0-2635, there is no noticeable reaction when Register 19 Bit 4 goes low, unless the GPS coding is customized to evoke a special reaction.

Troubleshooting Tip:

If a Fieldbus card is not found on the system, the Error Reaction setting will be ignored. If you have a Fieldbus card and the Error Reaction is not responding as expected, the system may not "see" your Fieldbus card.

4 Information for the PLC Programmer

4.1 *.gsd File

Indramat supplies a *.gsd file containing supporting information for the CLC-D with DPF05.x card Profibus slave configuration. Contact an Indramat technical representative for location of this file.

4.2 Multiplex Data Bits in the Control and Status Words

Figure 4-1: Bit Definitions in Multiplex Control and Status Words contains diagrams of the bits contained in the control and status words and their bit definitions. The bits marked with ✖ are currently not used. The control word is appended as the last word of the cyclic data content, and the status word is appended as the slave response in the cyclic data input.

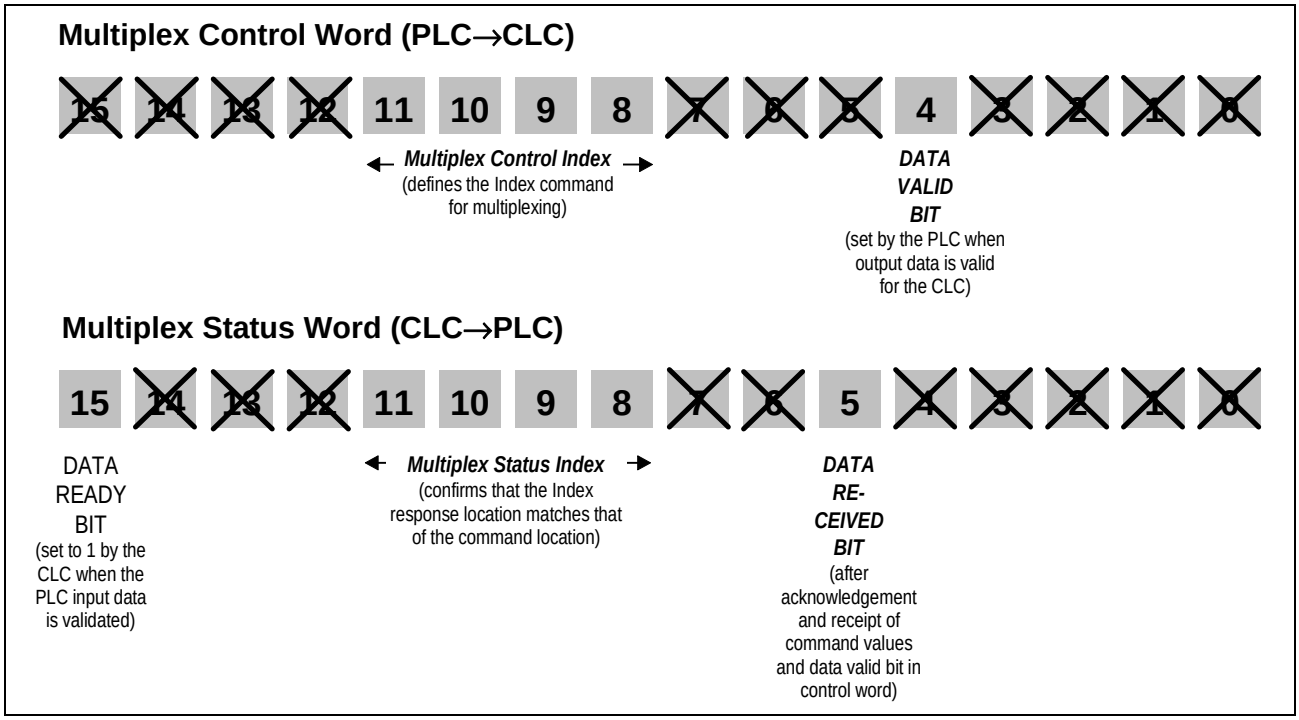

Multiplex Control Word (PLC→CLC)
15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
Multiplex Control Index
(defines the Index command for multiplexing)
DATA VALID BIT
(set by the PLC when output data is valid for the CLC)
Multiplex Status Word (CLC→PLC)
15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
DATA READY BIT
(set to 1 by the CLC when the PLC input data is validated)
Multiplex Status Index
(confirms that the Index response location matches that of the command location)
DATA RE-CEIVED BIT
(after acknowledgement and receipt of command values and data valid bit in control word)

Figure 4-1: Bit Definitions in Multiplex Control and Status Words

Figure 4-2: Write / Write and Read Process and ***Figure 4-3: Read-Only Process*** explain the exchange of multiplex information between the master (PLC) and the slave (CLC-D) using the control and status words:

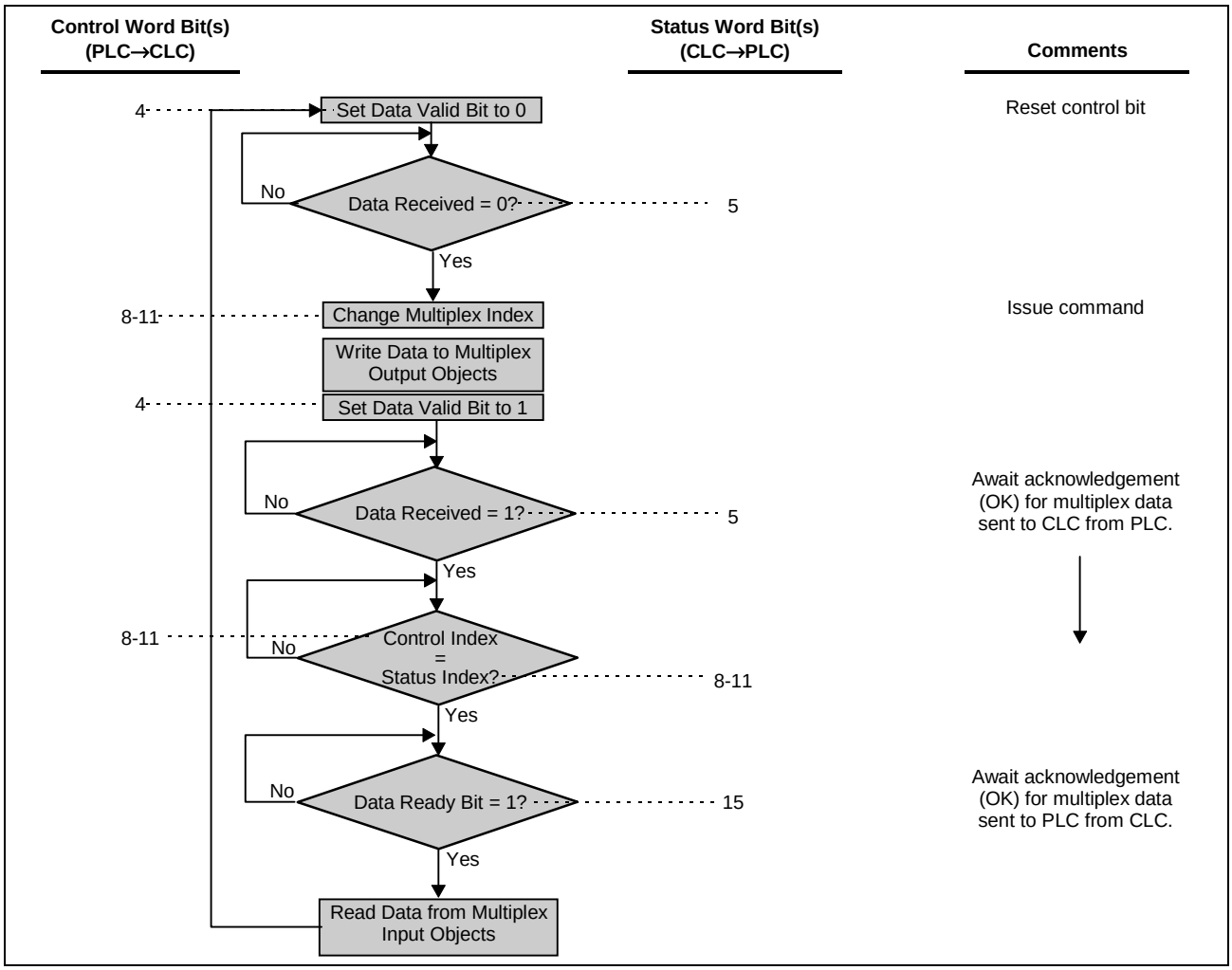


Figure 4-2: Write / Write and Read Process

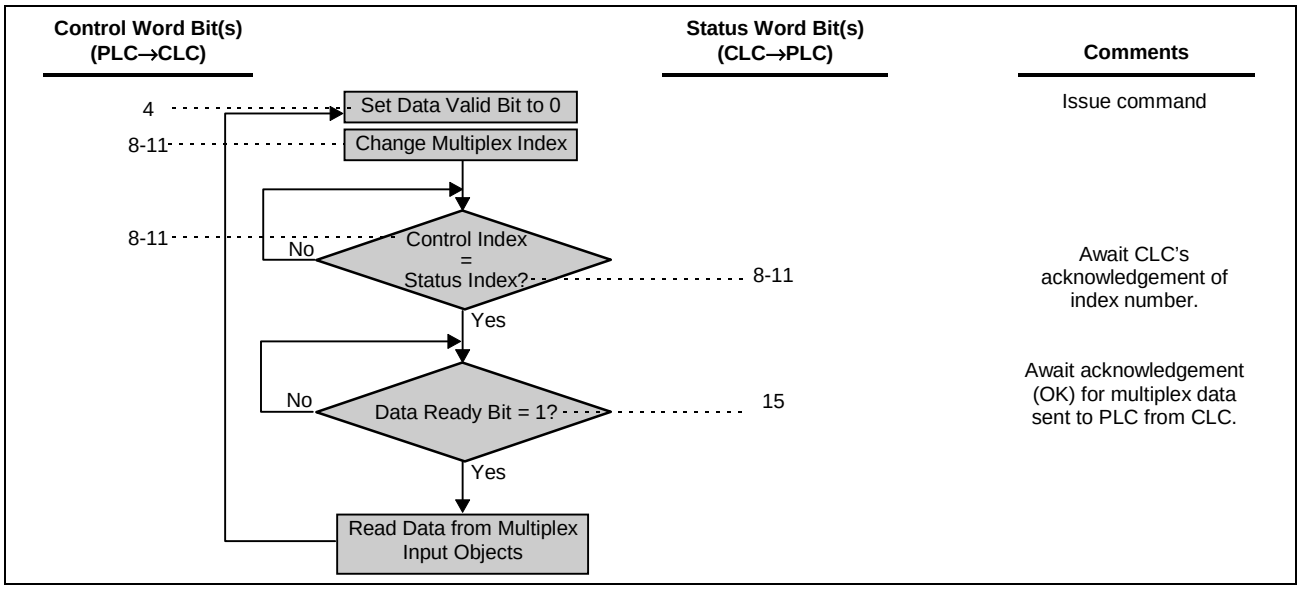


Figure 4-3: Read-Only Process

4.3 Parameter Channel in the Cyclic Data Channel (Process Data Channel on Profibus DPF05.x only)

Many PLC applications exclusively use the Profibus cyclic (DP) services. This means that parametrization via the non-cyclic channel (FMS) is not possible. There is, however, always the need to parametrize controls and drive via the Fieldbus. The DPF05.x board is therefore equipped with a configurable parameter channel, which allows the transmission of some non-cyclic data via the cyclic (DP) channel.

If this feature is desired, the first 4 or 6 data words of the cyclic channel for the slave board must be allocated for non-cyclic transmissions. This portion of the cyclic channel is subdivided as the parameter channel. The remainder of the cyclic channel is called the real-time channel (see ***Figure 4-4: The Non-Cyclic (DP) Channel*** below).

Note: The parameter channel is always allocated as the first 4 or 6 words of the Profibus cyclic (DP) channel. The length of the parameter channel plus the length of the real-time channel used to exchange cyclic data represent the entire length of the DP channel (maximum total length: 16 words).

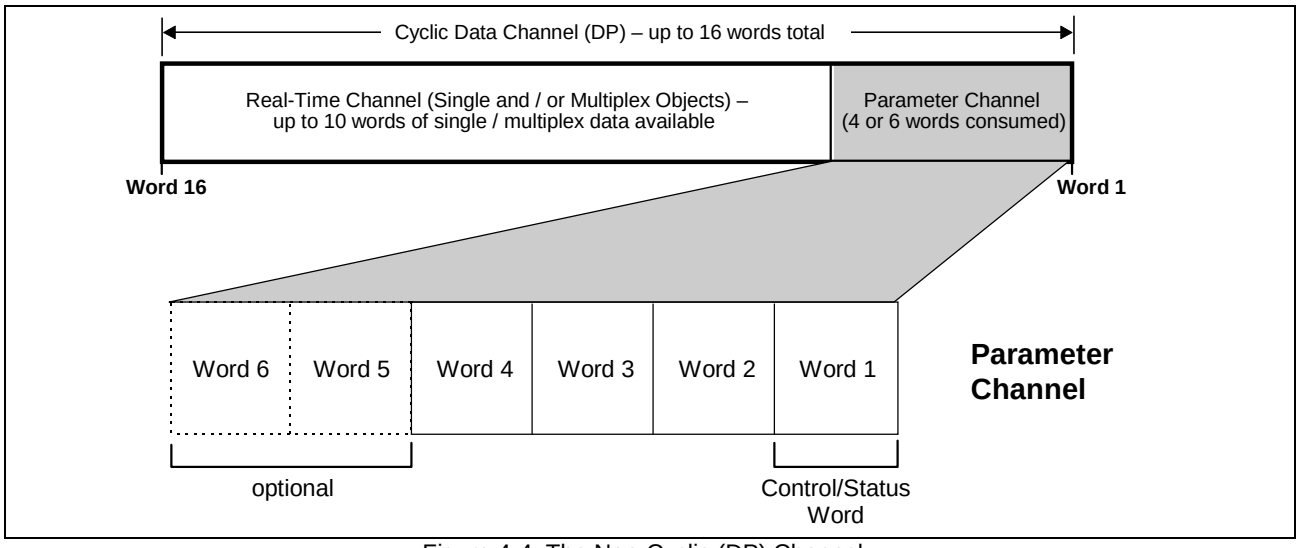

Cyclic Data Channel (DP) – up to 16 words total
Real-Time Channel (Single and / or Multiplex Objects) – up to 10 words of single / multiplex data available
Parameter Channel (4 or 6 words consumed)
Word 16
Word 1
Word 6
Word 5
Word 4
Word 3
Word 2
Word 1
Parameter Channel
optional
Control/Status Word

Figure 4-4: The Non-Cyclic (DP) Channel

Note: The total defined length of the cyclic (DP) channel must include the 4 or 6 data words for the parameter channel.

P-CHAN (Parameter Channel) Component Descriptions

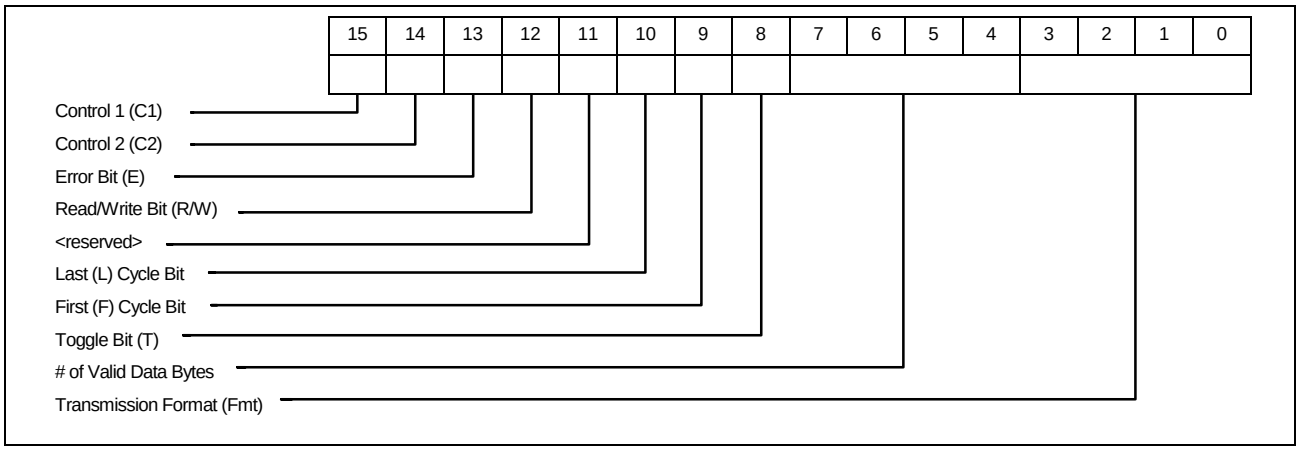

15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
Control 1 (C1)
Control 2 (C2)
Error Bit (E)
Read/Write Bit (R/W)
<reserved>
Last (L) Cycle Bit
First (F) Cycle Bit
Toggle Bit (T)
# of Valid Data Bytes
Transmission Format (Fmt)

Figure 4-5: Parameter Channel Control/Status Word Bit Descriptions

Bit Definitions of the Fieldbus Control / Status Word:

C1, C2 Control Bits Bits 15 and 14

These special control bits allow the reading of some of the FB slave setup information when used in conjunction with the R/W bit. In Read Mode (R/W = 1), it is possible to read the following:

- length of the cyclic channel
- length of the parameter channel (first 4-6 words of the cyclic channel)
- Fieldbus address

The following table lists the possible settings for these control bits:

Description	Bit 15 (C1)	Bit 14 (C2)
Reading the length of the cyclic data channel (parameter channel length + real-time channel length)	1	0
Reading the length of the parameter channel	1	1
Reading the fieldbus slave address	0	1
End reading of control bit information	0	0

Table 4-10: Control Bit Settings, Bits 15 and 14

Note: The slave response telegram will contain only the Fieldbus status word. The requested values are available in the low byte of the Fieldbus status word. (i.e. <#Bytes> + <Fmt> sections will contain the relevant information).

Error Bit (E) Bit 13

Diagnosing Parameter Channel Errors:

If the Fieldbus slave (DPF) detects an error during a data transfer via the parameter channel, the error bit #13 (E) in the Fieldbus status word will be set to 1. No error code is given for Short Format 2 or VisualMotion ASCII.

Clearing Parameter Channel Errors:

To clear a parameter channel error, the master sets the error bit to 1 and clears the error in the slave (if possible).

The master must then output the basic state in the control word. This **must** be done before the next parameter channel transmission can take place.

The basic state of the FB control word is 0x000X.

F = first cycle; L = last cycle Bits 10 and 9

Some data transactions (e.g. VisualMotion ASCII Format) may require multiple transmissions to complete a message. To detect a clear sequence for data transmission, the first and last cycles are accompanied

by a signal. This clearly identifies data transmissions extended over several cycles.

Transmission type	Bit 10 (L)	Bit 9 (F)
First transmission	0	1
Last transmission	1	0
Intermediate step	0	0
First and last transmission (all in one transmission)	1	1

Table 4-11: Control Bit Settings, Bits 10 and 9

Note: Independent of the type of transmission format, the master must identify the first and final data cycle. This is necessary to clarify the precise length of the data and specify the beginning and the end of a complete transmission.

Toggle Bit (T) Bit 8

To identify a new transmission step and confirm its receipt by the slave, a toggle bit is used in the Fieldbus control and status words. This bit provides a means of identifying data consistency over one or many transmissions of a transaction.

The functionality of the toggle bit is as follows:

1. The master changes the toggle bit with each data transmission cycle.
2. Upon receipt of data, the slave sets the toggle bit to the state specified by the master.
3. Only now may the master start up a new transmission cycle in which it can change more data and the toggle bit.

Exception: The master sets the basic state to clear an error.

Whether or not the toggle bit was previously set, the master recognizes the basic state by the coding of the parameter channel control word basic state: 0x000X.

Number of Valid Data Bytes (#Bytes) Bits 7-4

The number of valid data bytes (all bytes after the control or status word) for the current transmission is specified here.

Note: This value is the number of bytes of data in the current transmission. If multiple transmissions are needed to complete a data transaction, each transmission specifies the quantity of data attached to that transmission only. If the quantity of data is too large to be transmitted during one cycle, the number of valid bytes specified in the last transmission could be different from that number in previous transmissions.

Transmission Format (Fmt) Bits 3-0

This key section of the control/status word determines which data format is being specified. It must be properly set **every** time a particular format is used.

The following table lists the possible transmission formats:

Format	Bit 3	Bit 2	Bit 1	Bit 0
<None> (e.g. C1 and C2 control bits, or basic state)	0	0	0	0
Short Format 2	1	1	0	1
VisualMotion ASCII Format	1	0	1	1

Table 4-12: Control Bit Settings, Bits 3-0

Messaging Formats

Two messaging formats are available in the parameter channel:

- Short Format 2 (for direct access to Fieldbus mapped objects
- VisualMotion ASCII Format

Either format can be used at any time, depending on the application requirements. These two formats allow different levels of functionality within the parameter channel.

Short Format 2: General Explanation

This format allows for direct access to the Fieldbus objects on the DPF board. (These objects can also be directly mapped to VisualMotion data types.) Short Format 2 is the easiest format to implement from the PLC side, but it is also the most limiting in the flexibility of access and the number of different VisualMotion data types.

Short Format 2 is the Parameter Channel's method of accessing non-cyclic direct-mapped objects. For an explanation of direct-mapped objects, see ***Direct-Mapped Data*** on page 1-7. For setup procedures, follow the instructions for ***STEP IV: Define Non-Cyclic Data Mapping Lists (Direct Mapping)*** on page 2-8.

Short format 2 can be used with the following parameters and values:

- A parameters (CLC Axis parameters, 32 bit – 2 words)
- C parameters (CLC C system parameters, 32 bit – 2 words)
- T parameters (CLC Task parameters, 32 bit – 2 words)
- FP values (CLC Floating Point data, 32 bit – 2 words, IEEE format)
- GP values (CLC Global Integer data, 32 bit – 2 words)
- HP values (CLC Global Floating Point data, 32 bit – 2 words, IEEE format)
- IP values (CLC Integer data, 32 bit – 2 words)
- RX values (CLC Register data in Hex format, 16 bit – 1 word)

Note: These VisualMotion data types must be mapped to Fieldbus objects via the Fieldbus Mapper software tool. Data should be mapped by selecting the “FMS/P-Chan" (in the proper direction as well – read and write access must be specified in the FB Mapper). This list of mapped objects is stored in CLC card parameter C-2700.

User Data Header – Object Number Word

In Short Format 2, the object number refers to the particular Fieldbus slave object that a VisualMotion data type is mapped to. This object allows for simple, indirect access to VisualMotion data types. The required objects can be obtained from the Profibus non-cyclic (FMS/P-Chan) channel data mapping list (data obtained from VisualMotion system list parameter C-0-2700, valid range: 0x5e74 to 0x5fff).

Note: It is important to note the mapping of the object in the Fieldbus Mapper. If a particular object is mapped as a “VisualMotion Read Object,” you **cannot write** to the object in Short Format 2. Conversely, if the object is mapped as a "VisualMotion Write Object," you **cannot read** to that object in Short Format 2.

User Data – High Word / Bytes 0 & 1

In Short Format 2, this data word represents the high word of any user data that is either sent in the command telegram or received in the response telegram.

The format of this data is dependent upon the data type being transmitted.

Note: If the data type being transmitted is only 16 bit, this word is not used and should be set to NULL.

User Data – Low Word / Bytes 2 & 3

In Short Format 2, this data word represents the low word of any user data that is either sent in the command telegram or received in the response telegram.

The format of this data is dependent upon the data type being transmitted.

Write Command to Fieldbus Objects:

Short Format 2: Structure of the Command Telegram

Word	Word 1	Word 2	Word 3	Word 4	Word 5	Word 6
Data	FB control word	Object number	High-word	Low-word	Not used	
Message Format	Fieldbus control word	User data header	User data			
Message Layout	Parameter Channel Message Header (Motorola Format)		Parameter Channel Message Data (Intel Format)			

Read Command to Fieldbus Objects:

Word	Word 1	Word 2	Word 3	Word 4	Word 5	Word 6
Data	FB control word	Object number	Not used			
Message Format	Fieldbus control word	User data header				
Message Layout	Parameter Channel Message Header (Motorola Format)					

Short Format 2: Structure of the Response Telegram

Three cases must be differentiated for the response telegram in Short Format 2:

a) error-free execution (master command telegram was a read)

b) error-free execution (master command telegram was a write)

c) faulty execution

a) Error-Free Read Response

For an error-free execution after a master read command (E=0 in the Fieldbus status word), the response in the parameter channel contains 4 words. The slave response transmission (slave to master) is used when the master requests data from the slave. If the parameter channel is set for 6 words, words 5 and 6 are not used.

Word	Word 1	Word 2	Word 3	Word 4	Word 5	Word 6
Data	FB status word (E=0)	Object number	High-word	Low-word	Not used	
Message Format	Fieldbus status word	User data header	User data			
Message Layout	Parameter Channel Message Header (Motorola Format)		Parameter Channel Message Data (Intel Format)			

b) Error-Free Write Response

For an error-free execution after a master write command (E=0 in the Fieldbus status word), only the status word is returned. The remaining 3 or 5 words in the parameter channel are not used (depending on whether the parameter channel is set to 4 or 6 words).

Word	Word 1	Word 2	Word 3	Word 4	Word 5	Word 6
Data	FB status word (E=0)	Not used				
Message Format	Fieldbus status word					
Message Layout	Parameter Channel Message Header (Motorola Format)					

c) Faulty Execution Response

For a faulty execution after a master write command (E=1 in the Fieldbus status word), only the status word is returned. The remaining 3 or 5 words in the parameter channel are not used (depending on whether the parameter channel is set to 4 or 6 words). No detailed explanation of the transmission error is given in Short Format 2.

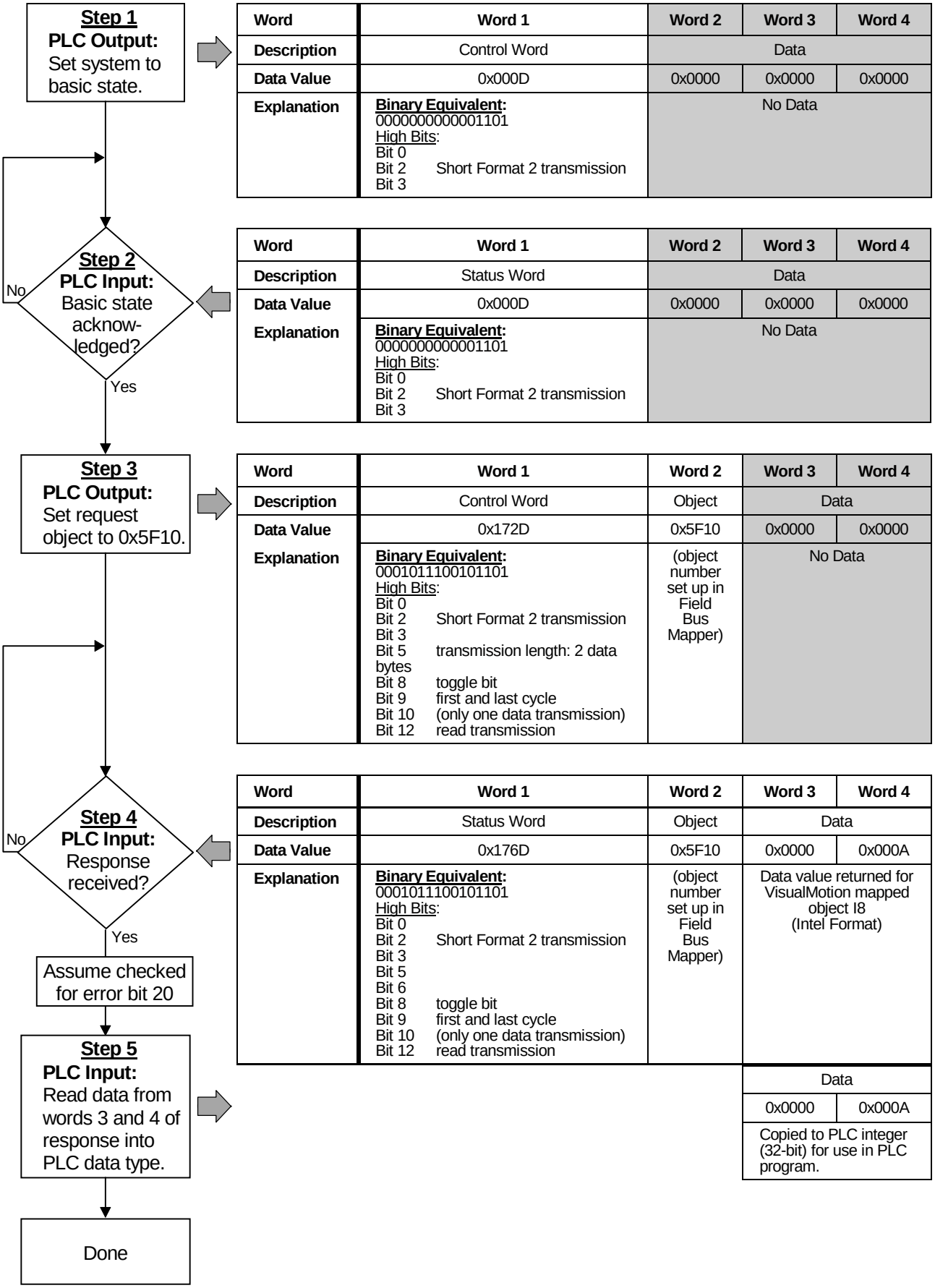
Word	Word 1	Word 2	Word 3	Word 4	Word 5	Word 6
Data	FB status word (E=1)	Not used				
Message Format	Fieldbus status word					
Message Layout	Parameter Channel Message Header (Motorola Format)					

Short Format 2: Examples

Example 1: Read Data

This example is for Short Format 2 only, with the Parameter Channel configured for 4 words (words 5 and 6, if configured, would not be used).

In this example, the transmission of VisualMotion Integer 8, which has been mapped in the VisualMotion Fieldbus Mapper, is illustrated. The current value of this integer is 10.



Word	Word 1	Word 2	Word 3	Word 4
Description	Control Word	Data		
Data Value	0x000D	0x0000	0x0000	0x0000
Explanation	**Binary Equivalent:** 0000000000001101 High Bits: Bit 0 Bit 2 Short Format 2 transmission Bit 3	No Data		

Word	Word 1	Word 2	Word 3	Word 4
Description	Status Word	Data		
Data Value	0x000D	0x0000	0x0000	0x0000
Explanation	**Binary Equivalent:** 0000000000001101 High Bits: Bit 0 Bit 2 Short Format 2 transmission Bit 3	No Data		

Word	Word 1	Word 2	Word 3	Word 4
Description	Control Word	Object	Data	
Data Value	0x172D	0x5F10	0x0000	0x0000
Explanation	**Binary Equivalent:** 0001011100101101 High Bits: Bit 0 Bit 2 Short Format 2 transmission Bit 3 Bit 5 transmission length: 2 data bytes Bit 8 toggle bit Bit 9 first and last cycle Bit 10 (only one data transmission) Bit 12 read transmission	(object number set up in Field Bus Mapper)	No Data	

Word	Word 1	Word 2	Word 3	Word 4
Description	Status Word	Object	Data	
Data Value	0x176D	0x5F10	0x0000	0x000A
Explanation	**Binary Equivalent:** 0001011100101101 High Bits: Bit 0 Bit 2 Short Format 2 transmission Bit 3 Bit 5 Bit 6 Bit 8 toggle bit Bit 9 first and last cycle Bit 10 (only one data transmission) Bit 12 read transmission	(object number set up in Field Bus Mapper)	Data value returned for VisualMotion mapped object I8 (Intel Format)	

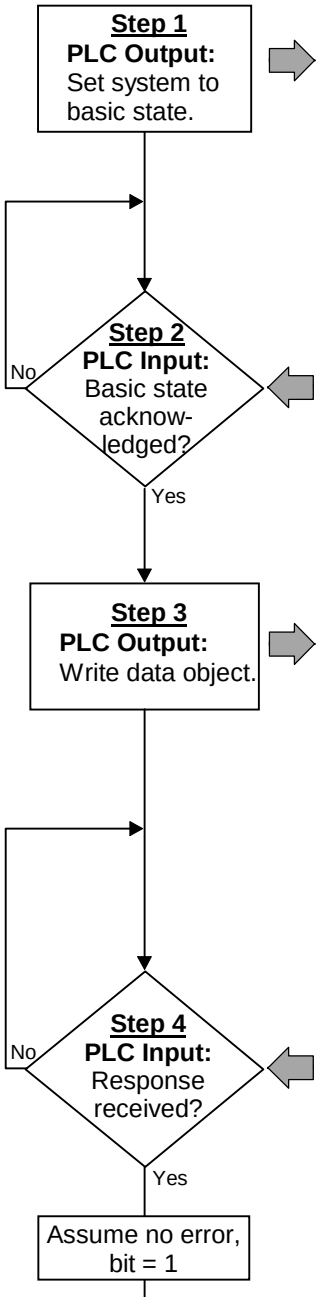
Data	
0x0000	0x000A
Copied to PLC integer (32-bit) for use in PLC program.	

Figure 4-6: Short Format 2: Read Example

Example 2: Write Data

This example is for Short Format 2 only, with the Parameter Channel configured for 4 words (words 5 and 6, if configured, would not be used).

In this example, data is written to object 5F00, which has been mapped to Float 16 in the VisualMotion Fieldbus Mapper. The current value of this float is 29.



Word	Word 1	Word 2	Word 3	Word 4
Description	Control Word	Data		
Data Value	0x000D	0x0000	0x0000	0x0000
Explanation	**Binary Equivalent:** 0000000000001101 High Bits: Bit 0, Bit 2, Bit 3 — Short Format 2 transmission	No Data		

Word	Word 1	Word 2	Word 3	Word 4
Description	Status Word	Data		
Data Value	0x000D	0x0000	0x0000	0x0000
Explanation	**Binary Equivalent:** 0000000000001101 High Bits: Bit 0, Bit 2, Bit 3 — Short Format 2 transmission	No Data		

Word	Word 1	Word 2	Word 3	Word 4
Description	Control Word	Object	Data	
Data Value	0x076D	0x5F00	0x0000	0x001D
Explanation	**Binary Equivalent:** 0000011100101101 High Bits: Bit 0, Bit 2, Bit 3 — Short Format 2 transmission Bit 5, Bit 6 — transmission length: 6 data bytes Bit 8 — toggle bit Bit 9, Bit 10 — first and last cycle (only one data transmission)	(object number set up in Field Bus Mapper)	Data value returned for VisualMotion mapped object F16 (Motorola format)	

Word	Word 1	Word 2	Word 3	Word 4
Description	Status Word	Data		
Data Value	0x070D	0x0000	0x0000	0x0000
Explanation	**Binary Equivalent:** 0001011100001101 High Bits: Bit 0, Bit 2, Bit 3 — Short Format 2 transmission Bit 8 — toggle bit Bit 9, Bit 10 — first and last cycle (only one data transmission)	No Data		

Figure 4-7: Write Transmission in Short Format 2

VisualMotion ASCII Format: General Explanation

The VisualMotion ASCII Format allows for the highest degree of flexibility with regard to data transmission, but it is the most complex format in terms of PLC implementation. When this format is received in the parameter channel, the entire message is transferred to the designated data exchange object and handled as if it were a non-cyclic (FMS) message. For an explanation of the data exchange object, see ***Data Exchange Objects*** on page 1-8. For PLC implementation, see ***Information for the PLC Programmer*** on page 4-1.

All VisualMotion data types except Fieldbus objects can be accessed using the VisualMotion ASCII Format.

This format is selected by specifying it the in the parameter channel control word. If parameters are transmitted in VisualMotion ASCII Format, the specifications outlined in the "Direct ASCII Communication" section of the VisualMotion 6.0 Reference Manual apply.

VisualMotion ASCII Protocol

In the VisualMotion ASCII format, this location (words 2-4 or words 2-6, depending on the length of the parameter channel) is where the appropriate VisualMotion ASCII protocol message would be inserted. For a detailed description of this protocol, please refer to Appendix A – “Direct ASCII Communication” in the VisualMotion GPS6.0 Reference Manual (DOK-VISMOT-VM*-06VRS**-FKB1-AE-P).

Note: Due to the nature of the VisualMotion ASCII protocol, many transactions are larger than the available 10 bytes per transmission. Therefore, it may be necessary to complete multiple transmissions in both the command and response telegrams to complete a transaction.

VisualMotion ASCII Format: Structure of the Command Telegram

Using the ASCII protocol to transmit data is a multi-transaction process (at least one to send the ASCII request / write, and at least one to get the response message from the VisualMotion System):

Write Message to CLC-D

1. Send the ASCII message to the CLC-D:

Each VisualMotion ASCII protocol must be transmitted with a write instruction in the parameter channel control word. The ASCII protocol determines whether the message sent is a request to read or write information. The interpretation of this request is handled by the VisualMotion system. The response to a successful command message is only the Fieldbus status word.

Word	Word 1	Word 2	Word 3	Word 4	Word 5	Word 6
Data	FB control word	ASCII bytes 1 and 2	ASCII bytes 3 and 4	ASCII bytes 5 and 6	ASCII bytes 7 and 8	ASCII bytes 9 and 10
Message Format	Fieldbus control word	VisualMotion ASCII Protocol (e.g. ">0 CP 0.5<CR><LF>")				
Message Layout	Parameter Channel Message Header (Motorola Format)	Parameter Channel Message Data (Intel Format)				
Note: Each ASCII character takes up 1 byte of memory space. If an ASCII protocol message to be transmitted is longer than 10 bytes, multiple transmissions are necessary to complete the transaction. Refer to the fieldbus control word for a description of how to handle multiple transmissions of a transaction. The largest ASCII message can be 128 bytes.						

Verify Status Word

2. Verify the status word.

Word	Word 1	Word 2	Word 3	Word 4	Word 5	Word 6
Data	FB status word	No Data				
Message Format	Fieldbus status word					
Message Layout	Parameter Channel Message Header (Motorola Format)					

Request Response from CLC-D

3. Request the response message from the CLC-D.

Word	Word 1	Word 2	Word 3	Word 4	Word 5	Word 6
Data	FB control word	No Data				
Message Format	Fieldbus control word					
Message Layout	Parameter Channel Message Header (Motorola Format)					

Receive Status Word, Grab Data

4. Receive the status word and, if O.K., grab the data.

Word	Word 1	Word 2	Word 3	Word 4	Word 5	Word 6
Data	FB status word	ASCII bytes 1 and 2	ASCII bytes 3 and 4	ASCII bytes 5 and 6	ASCII bytes 7 and 8	ASCII bytes 9 and 10
Message Format	Fieldbus status word	VisualMotion ASCII Protocol (e.g. ">0 CP 0.5 … …>")				
Message Layout	Parameter Channel Message Header (Motorola Format)	Parameter Channel Message Data (Intel Format)				
Note:	Each ASCII character takes up 1 byte of memory space. If an ASCII protocol message to be transmitted is longer than 10 bytes, multiple transmissions are necessary to complete the transaction. Refer to the fieldbus control word for a description of how to handle multiple transmissions of a transaction. The largest ASCII message can be 128 bytes.					

VisualMotion ASCII Format: Structure of the Response Telegram

Three cases must be differentiated for the response telegram in VisualMotion ASCII Format:

a) error-free execution (master command telegram was a write, as in Step 1 above)

b) error-free execution (master command telegram was a read, as in Step 3 above)

c) faulty execution

a) Error-Free Write Response

An error-free execution (E=0 in the Fieldbus status word) after a master write command means that only the Fieldbus status word is returned. This means that the command to send the message was acknowledged. The remaining 5 words in the Parameter Channel will not be used.

Word	Word 1	Word 2	Word 3	Word 4	Word 5	Word 6
Data	FB status word (E=0)	Not used				
Message Format	Fieldbus status word					
Message Layout	Parameter Channel Message Header (Motorola Format)					

b) Error-Free Read Response

An error-free execution (E=0 in the Fieldbus status word) after a master read command has a parameter channel structure of up to 6 words or it may span multiple transmissions to complete the transaction (if the ASCII message is longer than 10 bytes).

Word	Word 1	Word 2	Word 3	Word 4	Word 5	Word 6
Data	FB status word (E=0)				>0_CP_0.10 0_27\CR\LF	
Message Format	Fieldbus status word	VisualMotion user data				
Message Layout	Parameter Channel Message Header (Motorola Format)	Parameter Channel Message Data (Intel Format)				

c) Faulty Execution Response

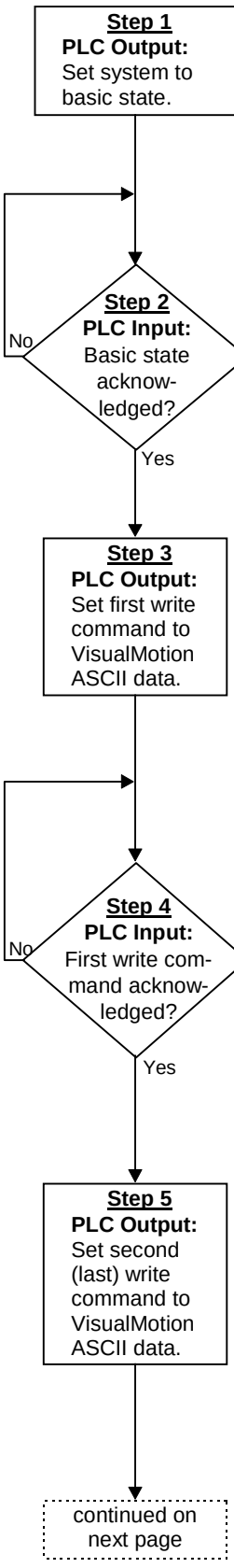
For a faulty execution of the write command process (E=1 in the Fieldbus status word), only the Fieldbus status word is returned. The remaining 5 words in the Parameter Channel will not be used. No detailed explanation of the transmission error is given in VisualMotion ASCII Format. The ASCII protocol, on the other hand, returns an error message if the format of the ASCII protocol is incorrect.

Word	Word 1	Word 2	Word 3	Word 4	Word 5	Word 6
Data	FB status word (E=1)	Not used				
Message Format	Fieldbus status word					
Message Layout	Parameter Channel Message Header (Motorola Format)					

VisualMotion ASCII Format: Example

This example is for VisualMotion ASCII Format, with the Parameter Channel configured for the maximum 6 words.

In this example, the value “0” is written to CLC-D Card Parameter 1.125 (the VisualMotion system timer).



Word	Word 1	Word2	Word3	Word4	Word5	Word6
Description	Control Word	Data				
Data Value	0x000B	0x0000	0x0000	0x0000	0x0000	0x0000
Explanation	**Binary Equivalent:** 0000000000001011 High Bits: Bit 0, Bit 1, Bit 3: VisualMotion ASCII Format	No Data				

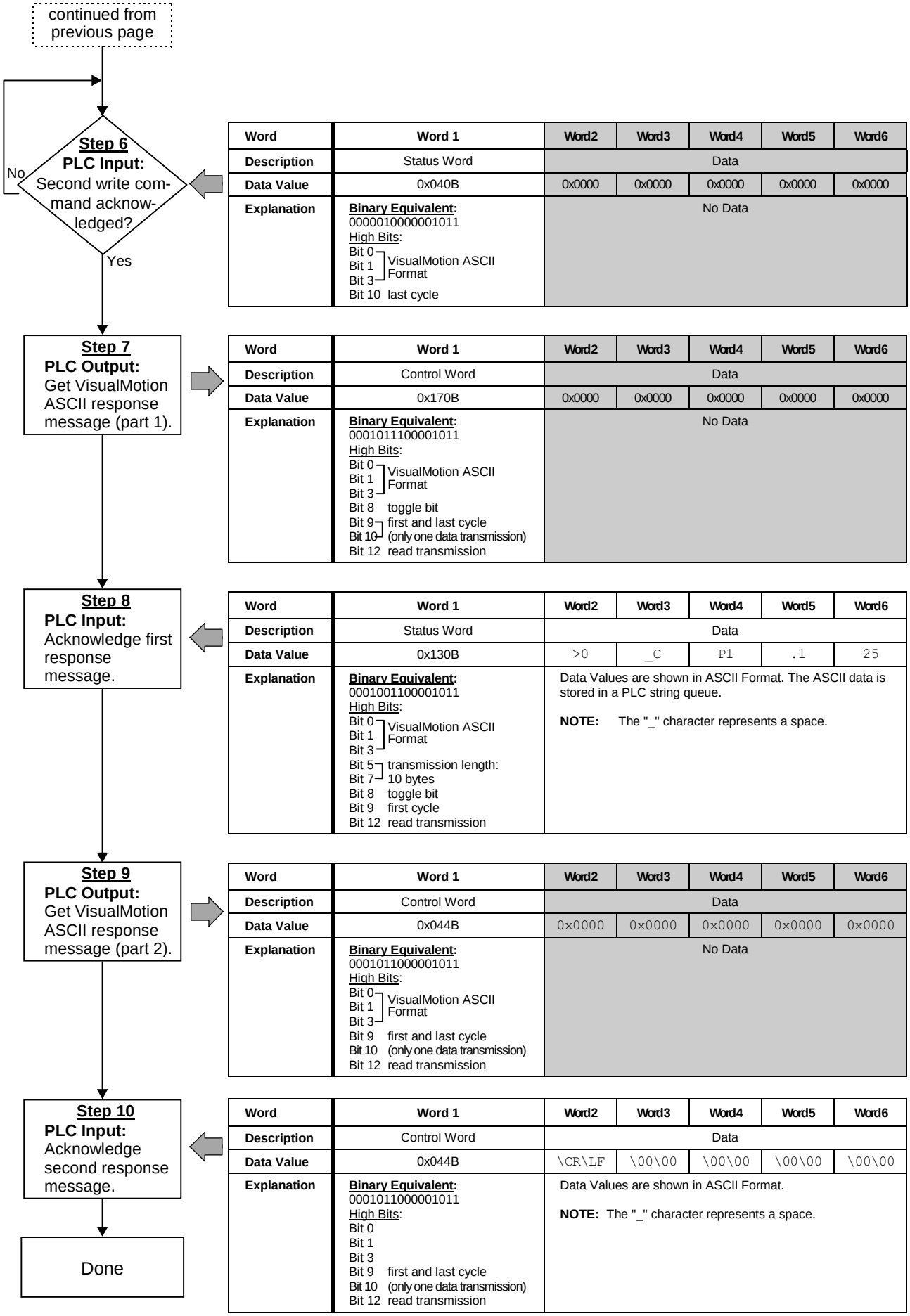
Word	Word 1	Word2	Word3	Word4	Word5	Word6
Description	Status Word	Data				
Data Value	0x000D	0x0000	0x0000	0x0000	0x0000	0x0000
Explanation	**Binary Equivalent:** 0000000000001011 High Bits: Bit 0, Bit 1, Bit 3: VisualMotion ASCII Format	No Data				

Word	Word 1	Word2	Word3	Word4	Word5	Word6
Description	Control Word	Data				
Data Value	0x03AB	>0	_C	P1	.1	25
Explanation	**Binary Equivalent:** 0000001110101011 High Bits: Bit 0, Bit 1, Bit 3: VisualMotion ASCII Format Bit 5, Bit 7: transmission length: 10 bytes Bit 8 toggle bit Bit 9 first cycle	Data Values are shown in ASCII Format. **NOTE:** The "_" character represents a space.				

Word	Word 1	Word2	Word3	Word4	Word5	Word6
Description	Status Word	Data				
Data Value	0x030B	0x0000	0x0000	0x0000	0x0000	0x0000
Explanation	**Binary Equivalent:** 0000001100001011 High Bits: Bit 0, Bit 1, Bit 3: VisualMotion ASCII Format Bit 8 toggle bit Bit 9 first cycle	No Data				

Word	Word 1	Word2	Word3	Word4	Word5	Word6
Description	Control Word	Data				
Data Value	0x044B	_0	\CR\LF	\00\00	\00\00	\00\00
Explanation	**Binary Equivalent:** 0000010001001011 High Bits: Bit 0, Bit 1, Bit 3: VisualMotion ASCII Format Bit 6 transmission length: 4 bytes Bit 10 last cycle **NOTE:** Bit 8 (toggle bit) is set low (0) in this step to reverse the toggle.	Data Values are shown in ASCII Format. **NOTE:** The "_" character represents a space. The sequence "\00" represents a "null" character.				

Figure 4-8: Data Transmission in VisualMotion ASCII Format (part 1 of 2)



Word	Word 1	Word2	Word3	Word4	Word5	Word6
Description	Status Word	Data				
Data Value	0x040B	0x0000	0x0000	0x0000	0x0000	0x0000
Explanation	**Binary Equivalent:** 0000010000001011 High Bits: Bit 0, Bit 1, Bit 3: VisualMotion ASCII Format Bit 10 last cycle	No Data				

Word	Word 1	Word2	Word3	Word4	Word5	Word6
Description	Control Word	Data				
Data Value	0x170B	0x0000	0x0000	0x0000	0x0000	0x0000
Explanation	**Binary Equivalent:** 0001011100001011 High Bits: Bit 0, Bit 1, Bit 3: VisualMotion ASCII Format Bit 8 toggle bit Bit 9, Bit 10: first and last cycle (only one data transmission) Bit 12 read transmission	No Data				

Word	Word 1	Word2	Word3	Word4	Word5	Word6
Description	Status Word	Data				
Data Value	0x130B	>0	_C	P1	.1	25
Explanation	**Binary Equivalent:** 0001001100001011 High Bits: Bit 0, Bit 1, Bit 3: VisualMotion ASCII Format Bit 5, Bit 7: transmission length: 10 bytes Bit 8 toggle bit Bit 9 first cycle Bit 12 read transmission	Data Values are shown in ASCII Format. The ASCII data is stored in a PLC string queue. **NOTE:** The "_" character represents a space.				

Word	Word 1	Word2	Word3	Word4	Word5	Word6
Description	Control Word	Data				
Data Value	0x044B	0x0000	0x0000	0x0000	0x0000	0x0000
Explanation	**Binary Equivalent:** 0001011000001011 High Bits: Bit 0, Bit 1, Bit 3: VisualMotion ASCII Format Bit 9 first and last cycle Bit 10 (only one data transmission) Bit 12 read transmission	No Data				

Word	Word 1	Word2	Word3	Word4	Word5	Word6
Description	Control Word	Data				
Data Value	0x044B	\CR\LF	\00\00	\00\00	\00\00	\00\00
Explanation	**Binary Equivalent:** 0001011000001011 High Bits: Bit 0 Bit 1 Bit 3 Bit 9 first and last cycle Bit 10 (only one data transmission) Bit 12 read transmission	Data Values are shown in ASCII Format. **NOTE:** The "_" character represents a space.				

Figure 4-9: Data Transmission in VisualMotion ASCII Format (part 2 of 2)

4.4 Non-Cyclic Transmission (Direct-Mapped Objects)

A number of objects (see ***Table 4-13: Non-Cyclic Data Objects*** below) can represent VisualMotion data types if they are mapped in the non-cyclic object mapping list (C-0-2700). This method provides a simple but inflexible method of transferring non-cyclic data from the master to the slave.

Number of Objects Available	Data Size	Direction	Numeric Names of Objects
16	32-bit	in	5F10-5F1F
16	32-bit	out	5F00-5F0F
32	16-bit	in	5F60-5F6F, 5FA0-5FAF
32	16-bit	out	5F70-5F7F, 5FB0-5FBF

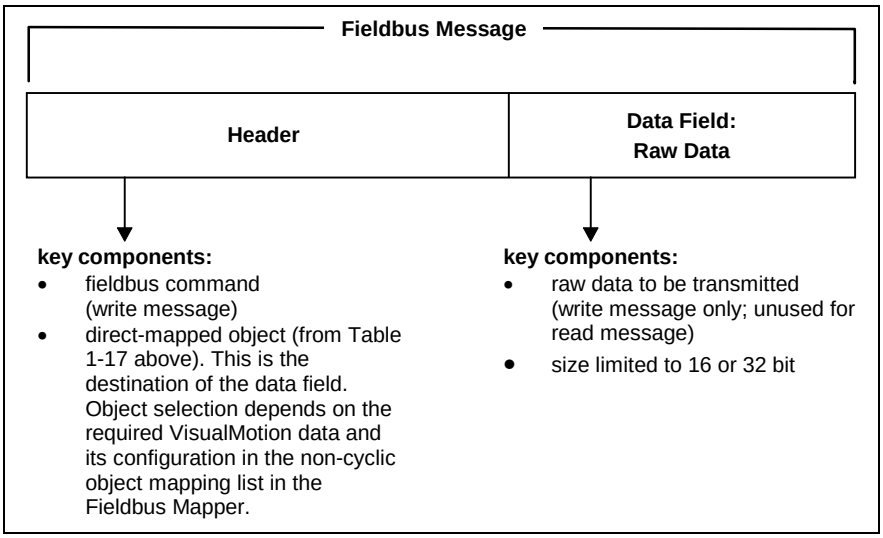
Table 4-13: Non-Cyclic Data Objects

Selecting a Direct-Mapped Object

The VisualMotion data available for non-cyclic direct-mapped objects can be viewed and printed via the summary report function in the VisualMotion Fieldbus Mapper tool (see ***Viewing/Printing a Summary Report of the Current Fieldbus*** on page 3-1).

Transmission Sequence via a Direct-Mapped Object

Note: For the direct-mapped object, only one transmission (and one response) is required, to send a read or write message to the CLC-D card and to receive a response from the CLC-D card.


Fieldbus Message
Header
Data Field:
Raw Data
key components:
• fieldbus command (write message)
• direct-mapped object (from Table 1-17 above). This is the destination of the data field. Object selection depends on the required VisualMotion data and its configuration in the non-cyclic object mapping list in the Fieldbus Mapper.
key components:
• raw data to be transmitted (write message only; unused for read message)
• size limited to 16 or 32 bit

Important: The format of the Fieldbus message header and the method of implementation are dependent on the Fieldbus type and the master (PLC) being used. Refer to your Fieldbus master/PLC documentation for proper transport and formatting of the message header.

Non-Cyclic Direct-Mapped Write

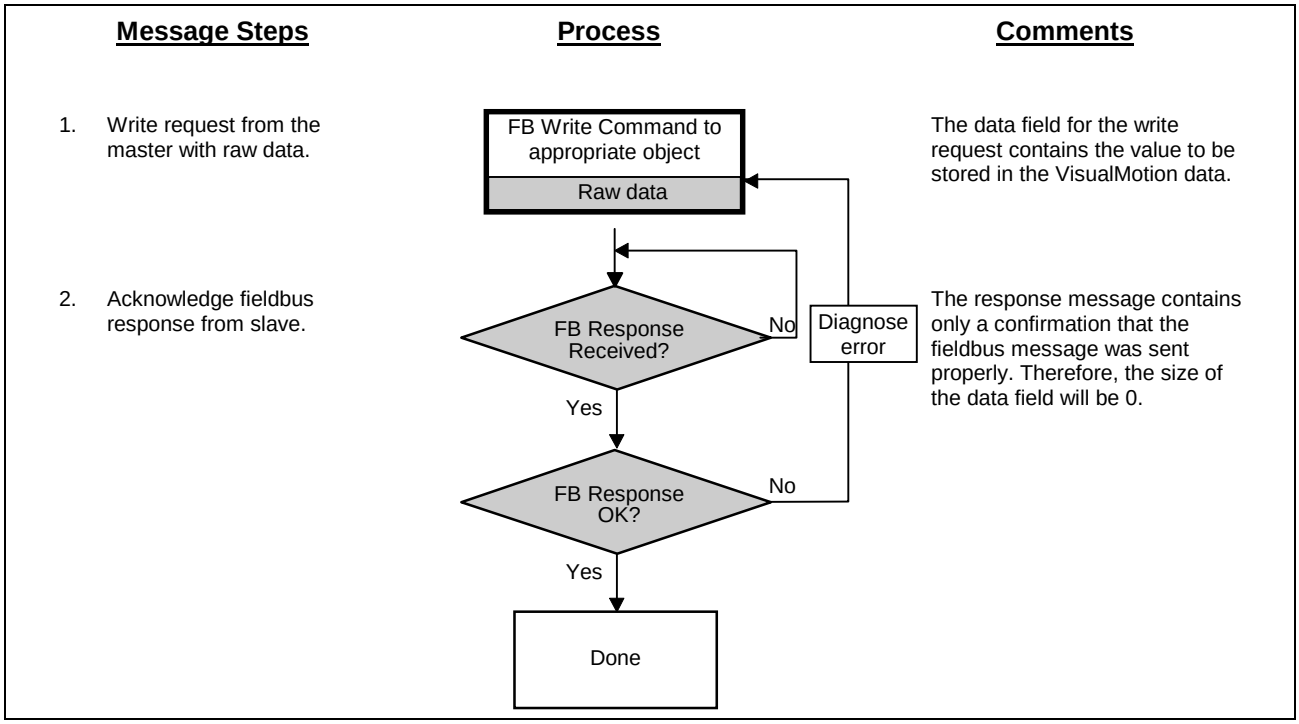
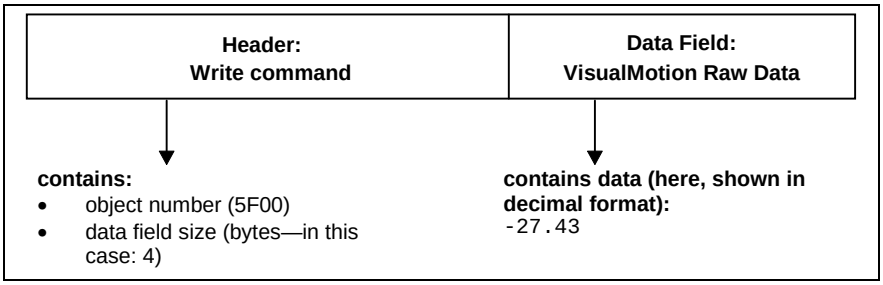

Message Steps
Process
Comments
1. Write request from the master with raw data.
FB Write Command to appropriate object
Raw data
The data field for the write request contains the value to be stored in the VisualMotion data.
2. Acknowledge fieldbus response from slave.
FB Response Received?
No
Diagnose error
The response message contains only a confirmation that the fieldbus message was sent properly. Therefore, the size of the data field will be 0.
Yes
FB Response OK?
No
Yes
Done

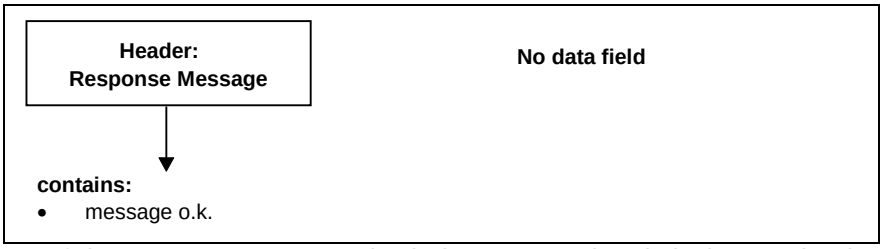
Figure 4-10: Non-Cyclic Direct-Mapped Write Process

Example: Write the value -27.43 to Float 16 (This is a 32-bit non-cyclic output object. In our Fieldbus Mapper example, it is mapped to object 5F00.)

1. Write request from the master with raw data.



2. After the write request from the master, the CLC-D card sends a response message.



3. If the message response (code in message header) shows o.k., the transaction is complete.

Note: It is important to note the mapping of the object in the Fieldbus Mapper. If a particular object is mapped as a “VisualMotion Read Object,” you **cannot write** to that object. Conversely, if the object is mapped as a "VisualMotion Write Object," you **cannot read** from that object.

Non-Cyclic Direct-Mapped Read

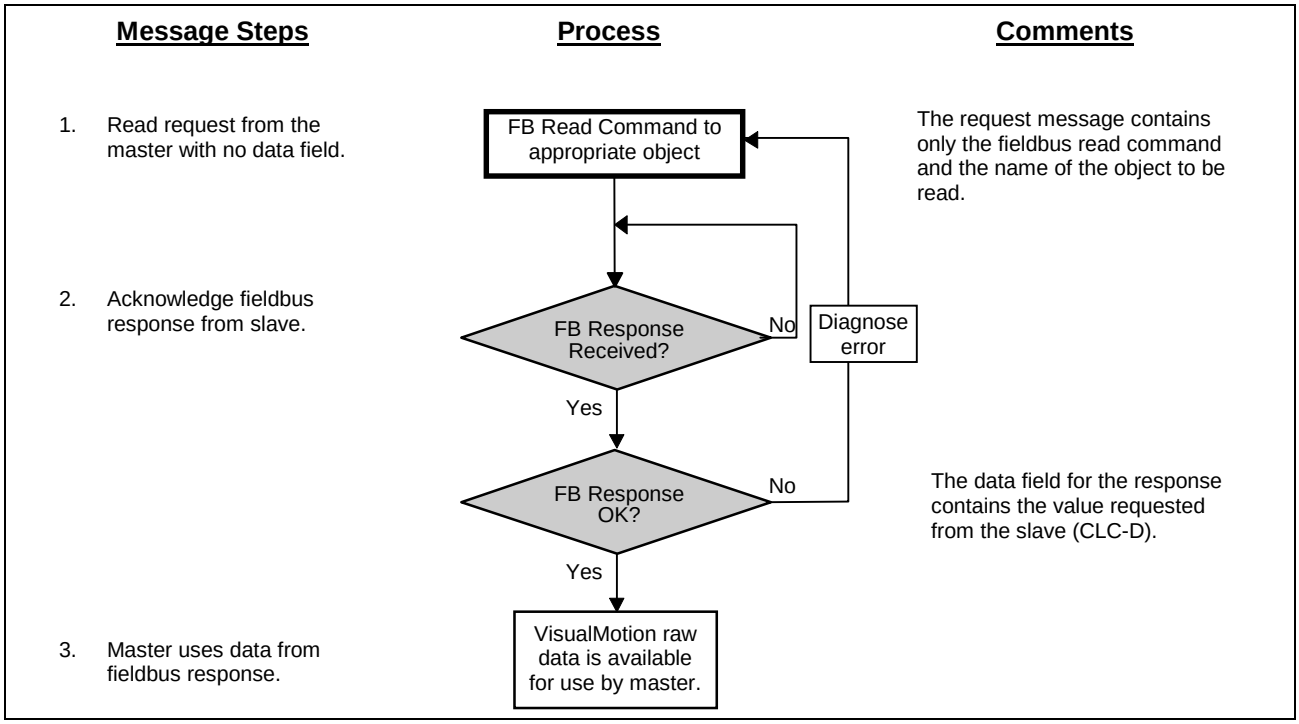
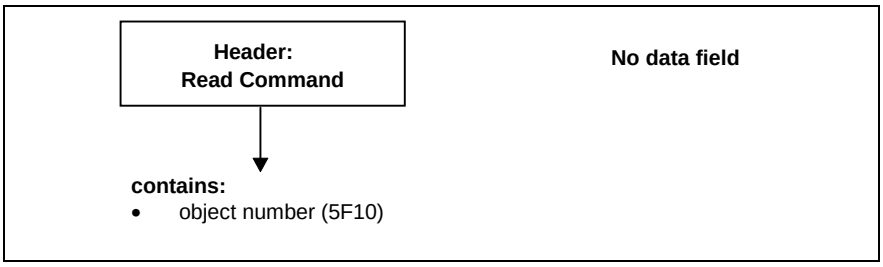

Message Steps
Process
Comments
1. Read request from the master with no data field.
FB Read Command to appropriate object
The request message contains only the fieldbus read command and the name of the object to be read.
2. Acknowledge fieldbus response from slave.
FB Response Received?
No
Diagnose error
Yes
FB Response OK?
No
The data field for the response contains the value requested from the slave (CLC-D).
Yes
3. Master uses data from fieldbus response.
VisualMotion raw data is available for use by master.

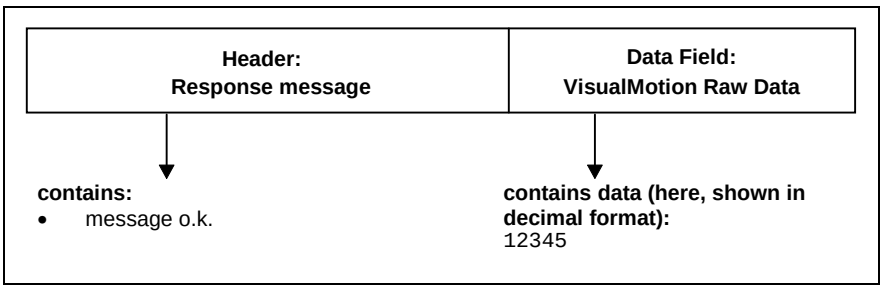
Figure 4-11: Non-Cyclic Direct-Mapped Read Process

Example: Read the value contained in Integer 8. (This is a 32-bit non-cyclic input object. In our Fieldbus Mapper example, it is mapped to object 5F10.)

1. Read request from the master.



2. After the read request from the master, the CLC-D card sends a response message.



If the message response (code in message header) shows o.k., the requested value is attached to the message in the data field. This value is now available for use by the master (PLC).

Note: It is important to note the mapping of the object in the Fieldbus Mapper. If a particular object is mapped as a “VisualMotion Read Object,” you **cannot write** to that object. Conversely, if the object is mapped as a "VisualMotion Write Object," you **cannot read** from that object.

4.5 Non-Cyclic Transmission (Data Exchange Objects)

The four data exchange objects 5E70 to 5E73 represent fixed data "containers" of varying lengths that transfer the VisualMotion ASCII Protocol to the CLC-D card. These objects serve as an open-ended possibility to access any VisualMotion data (including cams, diagnostic text, etc.), but more work is required in the master to perform a transmission of this type. Both the VisualMotion ASCII message and the Fieldbus transfer message must be formulated.

Table 4-1: Length of the Data Exchange Objects lists the available data exchange objects and their sizes.

Data Exchange Object	Data Length (in bytes)
5E70	16
5E71	32
5E72	64
5E73	128

Table 4-1: Length of the Data Exchange Objects

Selecting a Data Exchange Object

Depending on the length of a VisualMotion ASCII message, any of these data exchange objects can be selected.

Note: The entire data length of the data exchange object must always be transmitted even if the VisualMotion ASCII message is shorter.
For example, if you want to transmit an ASCII message of 42 bytes, you must use object 5E72. To avoid a response error from the Fieldbus slave, you must append 22 "Null" characters to the end of the ASCII message to complete a data size of 64 bytes.

Note: The checksum for the VisualMotion ASCII protocol is NOT used with the data exchange object. If the checksum is sent as part of the string, it will be ignored, and no checksum will be sent in the VisualMotion ASCII response messages. To ensure data integrity, the Fieldbus protocols support a low-level checksum.

Transmission Sequence via a Data Exchange Object

Note: For the data exchange object, two transmissions (and two responses) are required, to send the read or write message to and then receive the response message from the CLC-D card.

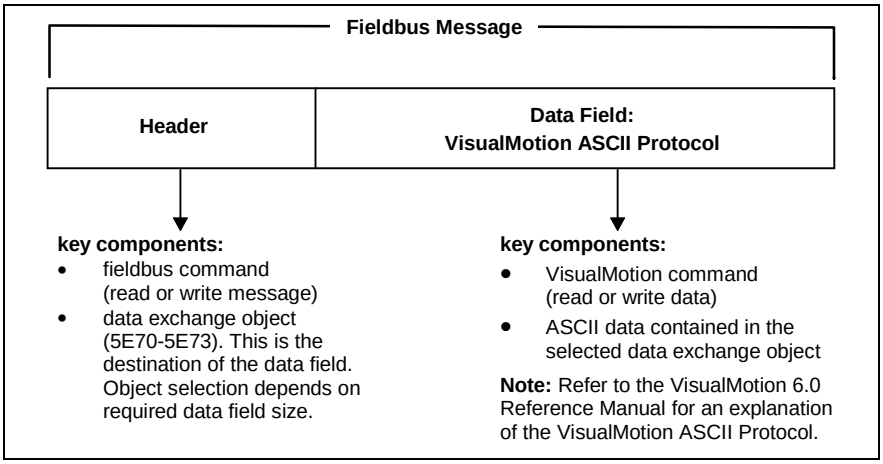

Fieldbus Message
Header
Data Field:
VisualMotion ASCII Protocol
key components:
• fieldbus command (read or write message)
• data exchange object (5E70-5E73). This is the destination of the data field. Object selection depends on required data field size.
key components:
• VisualMotion command (read or write data)
• ASCII data contained in the selected data exchange object
Note: Refer to the VisualMotion 6.0 Reference Manual for an explanation of the VisualMotion ASCII Protocol.

Figure 4-12: Format of a Non-Cyclic Fieldbus Message using a Data Exchange Object

Important: The format of the Fieldbus message header is dependent on the type of master (PLC) being used. Refer to your PLC manufacturer's manual for specific information on this topic.

The following sequence describes the communication between the Fieldbus master (PLC) and the Fieldbus slave (CLC-D):

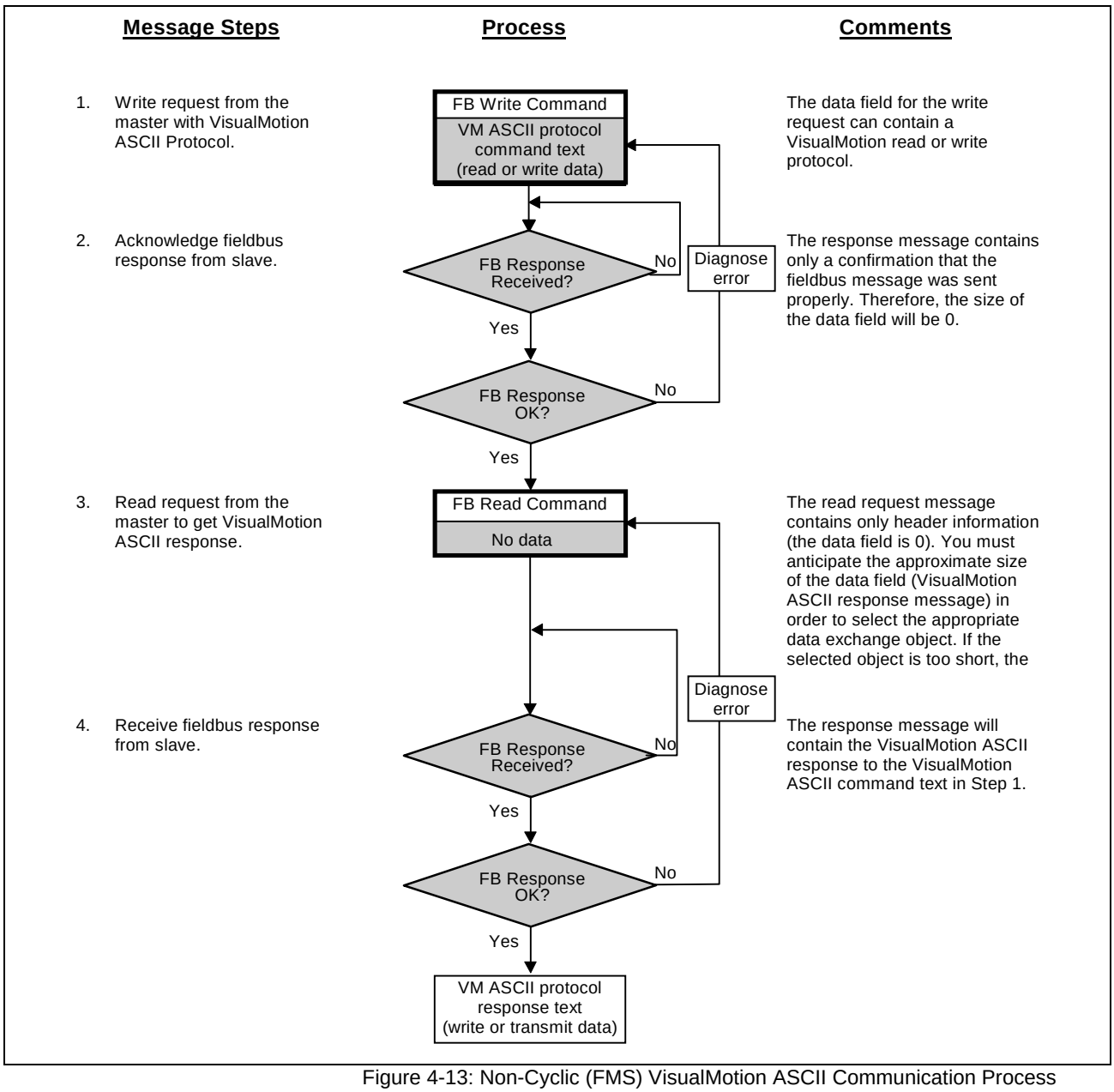
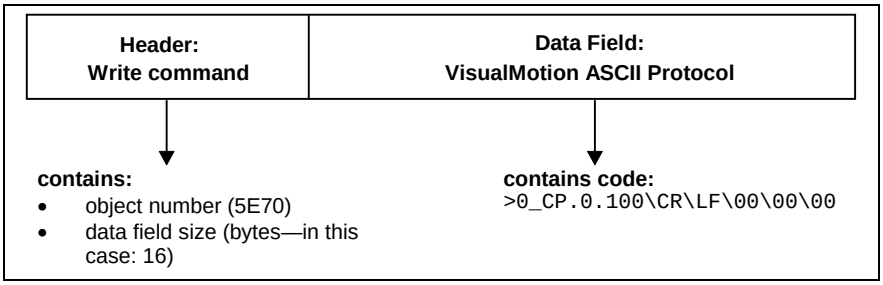

Message Steps
Process
Comments
1. Write request from the master with VisualMotion ASCII Protocol.
FB Write Command
VM ASCII protocol command text (read or write data)
The data field for the write request can contain a VisualMotion read or write protocol.
2. Acknowledge fieldbus response from slave.
FB Response Received?
No
Diagnose error
Yes
FB Response OK?
No
Yes
The response message contains only a confirmation that the fieldbus message was sent properly. Therefore, the size of the data field will be 0.
3. Read request from the master to get VisualMotion ASCII response.
FB Read Command
No data
The read request message contains only header information (the data field is 0). You must anticipate the approximate size of the data field (VisualMotion ASCII response message) in order to select the appropriate data exchange object. If the selected object is too short, the
4. Receive fieldbus response from slave.
Diagnose error
FB Response Received?
No
Yes
FB Response OK?
No
Yes
The response message will contain the VisualMotion ASCII response to the VisualMotion ASCII command text in Step 1.
VM ASCII protocol response text (write or transmit data)

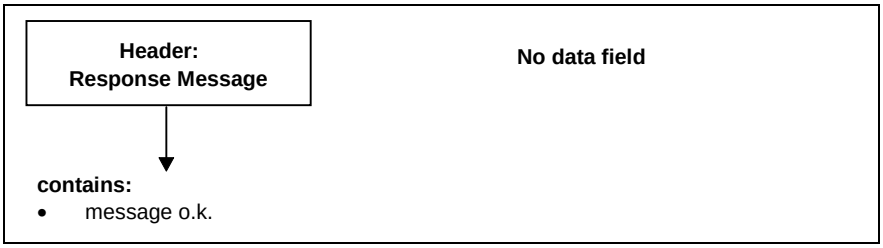
Figure 4-13: Non-Cyclic (FMS) VisualMotion ASCII Communication Process

Example: Read Card Parameter 100 (CLC-D firmware version)

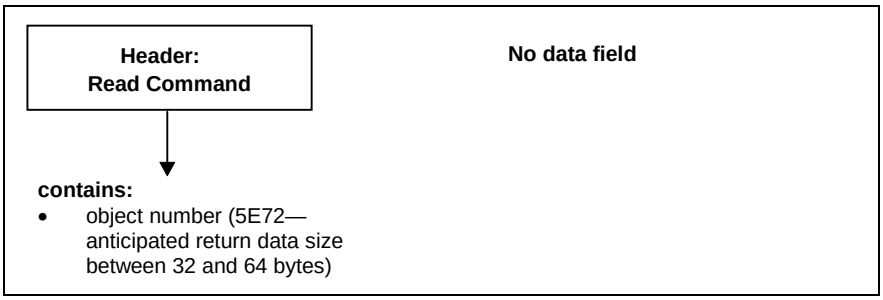
1. Write request from the master with VisualMotion ASCII Protocol.



2. After the first read request from the master, the CLC-D card sends a response message.

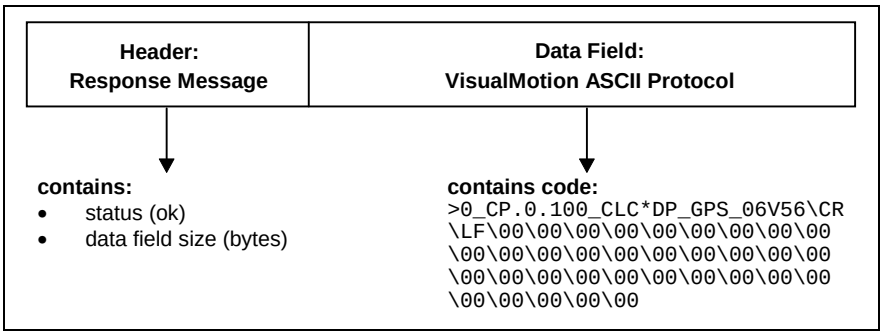


3. Read request from the master for the VisualMotion ASCII response message.



Note: To ensure that all of the data requested in this step is received in step 4 below, a data exchange object of the appropriate size must be selected.
If the selected data exchange object is too small, the data will be truncated.
If the selected data exchange object is too large, efficiency of transmission will be compromised.

4. The CLC-D sends the final response message.



5 Profibus DP Combi Slave Board DPF05.x

5.1 Application

The DPF05.x board enables the integration of the CLC-D control board into the Profibus system. Because the board is a Profibus DP combi slave board, integration into the following network types (individually or combined according to DIN 19245-3) is possible:

- Profibus - DP
- Profibus - FMS

Using this board, real-time data via the process data channel (requires Profibus DP), as well as parameters and data via FMS-specified objects can be transmitted, as long as the network is additionally or alternately supported as a Profibus FMS service.

Note: The use of a master scanner which supports both Profibus-DP and Profibus-FMS is recommended.

5.2 Functional Description

The DPF05.x board has the following functional features:

- Profibus cyclic (DP) and non-cyclic (FMS) channels are both supported
- all data supported according to DIN 19245-3 to 12 MBit
- instrinsic microcontroller for autonomous processing of FMS and DP functions
- monitoring of DP channel (watchdog function), with data exchange to the CLC-D card via dual-port RAM
- hardware and software synchronization with the CLC-D control card
- diagnostic LEDs on the front panel of the DPF05.x board for easy diagnosis of the bus functions and the communication between the DPF05.x and CLC-D cards
- implementation of an object structure for simplified accessing of variables and parameters on the CLC-D control card and drives
- upload/download function via 4 data exchange objects (16 to 128 bytes in data length)

5.3 Profibus Interface

The Profibus-DP combi slave board DPF05.x supports:

- interfaces according to DIN 19245, Sections 1 and 2
- expanded interface according to DIN 19245, Section 3
- line types A and B according to DIN 19245, Section 3

Note: DPF05.x supports baud rates of 9.6 to 12000 KBit.

To guarantee EN standard for EMC safety the Profibus interface is completely optically isolated.

According to DIN 19245, Section 1, the DPF05.x board has a 9-pin D-sub miniature connector for connection to the Profibus.

To switch the bus signal through to the rest of the bus participants, plug-in connector INS 0450 is available.

5.4 DPF05.x Board Hardware

Front view of the DPF05.x

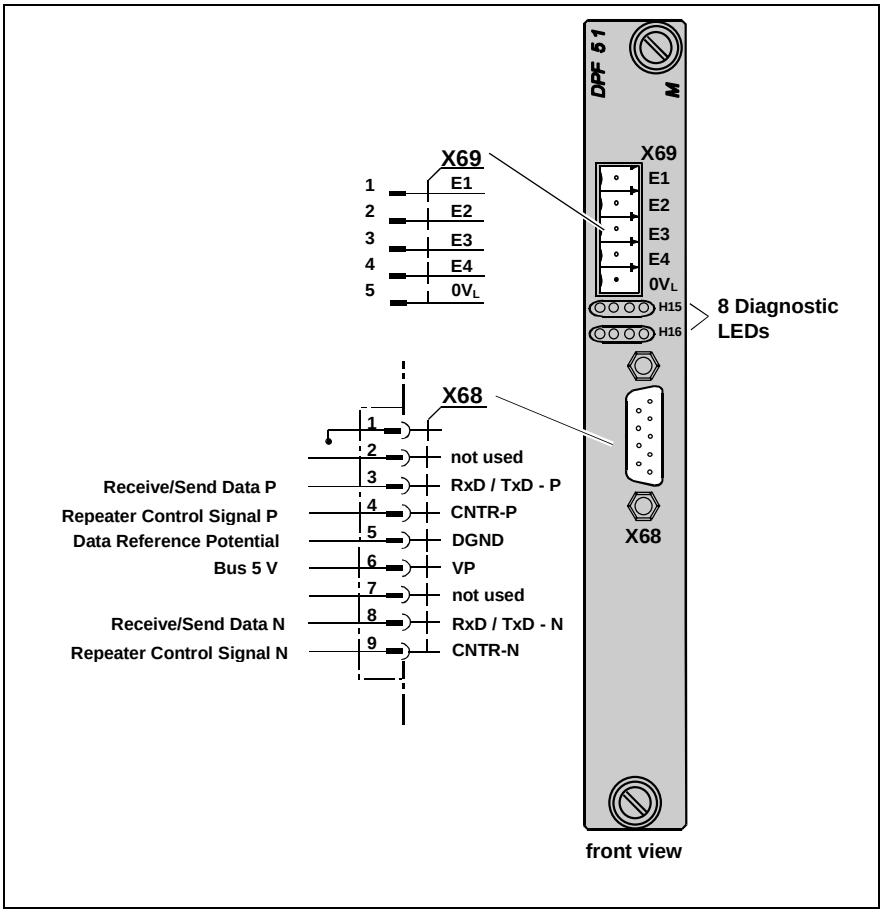


Figure 5-1: Front View of the DPF05.x

DPF05.x Structure

The DPF05.x board has been designed as a plug-in board so that it can be directly mounted to the CLC-D control board. Once secured with three guide pins, it forms one unit with the CLC-D card, which can be inserted into two slots on the drive or CCD card rack.

Note: Additional boards can be attached to the DPF05.x card for integration into the system. Exercise caution when disassembling or removing the card rack.

Power (+5V) is supplied by the drive or CCD card rack via a connector on the back of the DPF05.x.

Note: The DBS03.x or DCF01.x board may not be operated along with a Profibus DPF05.x slave board. Only one DPF05.x board can be connected to the CLC-D at a time.

The DPF05.x has the following interfaces:

Interface to the Drive or CCD Card Rack

This interface is used to supply power to the DPF05.x board if the CLC-D card does not supply the power.

Interface between CLC-D and Other Card Racks

Information is exchanged between the CLC-D and other card racks via this interface. The interface is an extended bus interface with partial coding of the address space so that other cards can also be mounted (e.g. DEA28, DEA29, DEA30, etc.).

External Inputs

There are three hardware outputs (+24V) on the DPF05.x. These inputs can only be used with the CLC-D if supported by the firmware. The signal states at this input are transmitted from the Profibus (on/off state) to the CLC-D. They can also be queried by the Profibus master via the PD or the FMS channel.

Note: The external inputs are not supported on the CLC-D card.

Profibus Interface

The Profibus interface meets requirements listed in DIN 19245, Section 1 for category A and B lines according to DIN 19245, Section 3.

Handheld Terminal

A handheld terminal (Siemens) can be connected via the Profibus interface to set addresses.

X68 Connector, Profibus Pin-outs

X 68	RS 485 Reference	INDRAMAT Signal Name	Signal per DIN 19245 Section 3	Definition
1		PE	shield	shield or protective ground
2		not used		
3	B / B´	B	RxD / TxD-P	receive/send data P [1]
4		CNTR-P	CNTR-P	repeater control signal P [2]
5	C / C´	BUSGND	DGND	data reference potential [1]
6		VP	VP	supply voltage plus (P5V) [2]
7		not used		
8	A / A´	A	RxD / TxD-N	receive / send data N [1]
9		CNTR-N	CNTR-N	repeater control signal N [2]

Notes:
[1] minimum received connections
[2] for power fail module only

Table 5-1: X68 Connector, Profibus Pin-outs

X69 Connector, External Inputs

Note: Not functional at this time.

X69	Label	Input Voltage High	Input Voltage Low
1	E1	+16 V... +32 V	-0.5 V ... +8 V
2	E2	+16 V... +32 V	-0.5 V ... +8 V
3	E3	+16 V... +32 V	-0.5 V ... +8 V
4	E4	+16 V... +32 V	-0.5 V ... +8 V
5	$0V_L$	reference potential 0V	reference potential 0V

Table 5-2: X69 Signal Configuration, External Inputs

DPF05.x Diagnostics

Front Panel LEDs

The DPF05.x front panel has eight diagnostic LEDs. They enable the diagnosis of the Profibus current status and the communication between the DPF05.x and CLC-D cards.

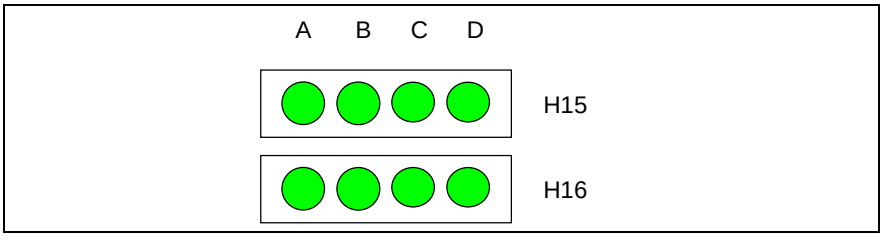


Figure 5-2: Arrangement of DPF05.x Board Diagnostic LEDs

Definition of Diagnostic LEDs

LED	Signal	Status	Definition
H15A	**USYS--Power source of Profibus board O.K.**	Green	Power is supplied to the PF05.x via the CLC-D card, the X4 plug-in connector or the X5 drive interface.
H15B	**BA--Profibus DP active**	Green	This LED is on if the Profibus cyclic channel (DP) is active. The cyclic data is continually monitored. If data transfer is detected during a Timeout, this LED turns on.
H15C	**TR--FMS transmission active**	Green	This LED is on while an FMS communication is exercised between the Profibus master and this slave.
H15D	**currently not used**	Green	--
H16A	**UBUS--P5V Profibus OK**	Green	This LED is on when the Profibus slave receives +5V. Simultaneously, line VP is applied to +5V pin 6 of the Profibus connector. A handheld terminal can be operated. This voltage is taken from the DC/DC converter on the DPF05.x board.
H16B	**SW-RUN--Software-RUN**	Green	This LED is used with H16C. It supports diagnosis of correctly running software and successful synchronization between the CLC-D and DPF05.x cards. If the CLC-D and DPF05.x cards are synchronized, the LEDs flash alternately.
H16C	**SW-RUN--Software-RUN**	Green	See description for LED H16B
H16D	**currently not used**	Green	

6 Index

*

*.gsd file....4-1
Basic Example.... 2-3
Multiplex Example.... 2-14

2

208 Lost Fieldbus Connection..3-10
209 Field Bus Mapper Timeout.3-10

5

519 Lost Fieldbus Connection..3-10
520 Field Bus Mapper Timeout.3-10
5E70.... 4-19
5E71.... 4-19
5E72.... 4-19
5E73.... 4-19

A

Available Objects
Basic Example....2-4
Multiplex Example
Cyclic Data....2-14

B

baud rates
supported by DPF05.x.... 5-1
Bit Definitions
Register 19
Cyclic Data Valid....3-7
FB Init OK....3-6
FB Slave Ready....3-6
nCyc Chan Ready....3-7
Non-Cyc Ready....3-7
nVM FW OK....3-7
Register 20
FB Card Fault Code....3-8
FB Card Found....3-8
Register 26
Current Miss Counter....3-9
Fieldbus Timeout Counter....3-9
Peak Miss Counter....3-9
Bus Conf. IN List
Basic Example.... 2-3
Multiplex Example
Cyclic Data....2-14
Bus Conf. OUT List
Basic Example.... 2-5
Multiplex Example
Cyclic Data....2-15
Bus Configuration Lists
APPEND Button.... 2-4, 2-14
DEL Button.... 2-4, 2-14
INSERT Button.... 2-4, 2-14
NEW Button.... 2-4, 2-14
Bus Type
Basic Example.... 2-2
Cyclic Data....2-6
Multiplex Example
Cyclic Data....2-12, 2-16
Non-Cyclic Data....2-18
Buttons
APPEND.... 2-4, 2-14
DEL.... 2-4, 2-14
Display Summary...2-7, 2-9, 2-18, 2-19
Exit....2-7, 2-9, 2-18, 2-19
Get From CLC....2-7, 2-9, 2-18, 2-19
Get From File....2-7, 2-9, 2-18, 2-19
INSERT....2-4, 2-14
NEW..... 2-4, 2-7, 2-9, 2-14, 2-18, 2-19
Save To File....2-7, 2-9, 2-18, 2-19
Send To CLC....2-7, 2-9, 2-18, 2-19

C

C-0-2600.... 2-7, 2-9, 2-18, 2-19
C-0-2600....1-2
C-0-2635....3-9
C-0-2700.... 1-7, 2-9, 2-18, 2-19
channels
Cyclic Data Channel configuration1-3, 1-6
multiplex.... 1-2
Non-Cyclic Channel.... 1-2
Parameter Channel.... 1-2
Real-Time Channel....1-2, 1-3
single.... 1-2
Clearing Parameter Channel Errors4-4
Command Telegram Structure
Short Format 2.... 4-7
VisualMotion ASCII Format.... 4-11
Configure Multiplex Channel
Basic Example.... 2-3
control word....1-5
Multiplex Example
Cyclic Data.... 2-14
Cyclic Channel....1-2
Cyclic Data Channel
Basic Example.... 2-6
configuration.... 1-3, 1-6
Multiplex Example.... 2-16
Cyclic Data Valid....3-7

D

Data Configuration List
Param.... 2-20
Data Exchange Objects.. 1-8, 4-19
5E70.... 4-19
5E70 to 5E73.... 1-8
5E71.... 4-19
5E72.... 4-19
5E73.... 4-19
Data Mapping
Direct-Mapped Data.... 1-7
Data Objects....1-7
Data Sizes
Cyclic.... 1-4
Data Type
Basic Example
Cyclic Data.... 2-6
Non-Cyclic Data.... 2-8
Multiplex Example
Cyclic Data.... 2-16
Non-Cyclic Data.... 2-18
Data Types
Cyclic.... 1-4
Multiplex....1-3, 1-4, 2-11
Single....1-3, 1-4
Define/Config Slave Objects
Basic Example.... 2-3
Multiplex Example
Cyclic Data.... 2-13
Destination

Basic Example
Cyclic Data 2-6
Non-Cyclic Data 2-8
Multiplex Example
Cyclic Data 2-16
Non-Cyclic Data 2-18
Device Address
Basic Example 2-3
Multiplex Example
Cyclic Data 2-13
Diagnostics
LEDs 5-3
Direct-Mapped Data 1-7
Display Summary
Cyclic, Multiplex Data 2-18
Cyclic, Single Data 2-7
File Menu entry 3-1
Non-Cyclic, Multiplex Data 2-19
Non-Cyclic, Single Data 2-9
DPF05.x
Hardware 5-2
Interfaces 5-2
DPF05.x board 1-1

E

EMC safety 5-1
Error Bit (E) 4-4
Error Reaction
Ignore 3-10
Shutdown CLC 3-10
Warning Only 3-10
Errors
208 Lost Fieldbus Connection 3-10
209 Field Bus Mapper Timeout 3-10
519 Lost Fieldbus Connection 3-10
520 Field Bus Mapper Timeout 3-10
Exit
Cyclic, Multiplex Data 2-18
Cyclic, Single Data 2-7
Non-Cyclic, Multiplex Data 2-19
Non-Cyclic, Single Data 2-9
External Inputs 5-3

F

FB Card Fault Code 3-8
FB Card Found 3-8
FB Init OK 3-6
FB Slave Ready 3-6
Field Bus Error Reaction
Basic Example 2-3
Multiplex Example
Cyclic Data 2-14
Field Bus Mapper
Examples
Multiplexing 2-11
Parameter Channel 2-20
Examples
Basic 2-1
Fieldbus Diagnostics 3-7
Fieldbus Error Reaction 3-9
Fieldbus Mapper 1-1
Fieldbus Master 1-1
fieldbus message header 4-16
Fieldbus Resource Monitor 3-8
Fieldbus Slave 1-1
Fieldbus Status 3-6
fieldbus summary report 2-7
Fieldbus-Accessible Parameters 3-3
File Menu
Display Summary 3-1
Print 3-2
first cycle 4-4
FMS (Fieldbus Message Specification) 1-2
From FieldBus
Basic Example 2-4
Multiplex Example
Cyclic Data 2-15

G

Get From CLC
Cyclic, Multiplex Data 2-18
Cyclic, Single Data 2-7
Non-Cyclic, Multiplex Data 2-19
Non-Cyclic, Single Data 2-9
Get From File
Cyclic, Multiplex Data 2-18
Cyclic, Single Data 2-7
Non-Cyclic, Multiplex Data 2-19
Non-Cyclic, Single Data 2-9
GPS Programming 3-1

H

H15A 5-4
H15B 5-4
H15C 5-4
H15D 5-4
H16A 5-4
H16B 5-4
H16C 5-4

I

Ignore (Fieldbus Error Reaction) 3-10
Index Number
Multiplex Example
Cyclic Data 2-14
input 1-2

L

last cycle 4-4
least significant bit 3-6
LEDs
H15A 5-4
H15B 5-4
H15C 5-4
H15D 5-4
H16A 5-4
H16B 5-4
H16C 5-4
LEDs (DPF05.x) 5-3
Length of DP Channel (Word)
Basic Example 2-3
Multiplex Example
Cyclic Data 2-13

M

Mapping Direction
Basic Example
Cyclic Data 2-6
Non-Cyclic Data 2-8
Multiplex Example
Cyclic Data 2-16
Non-Cyclic Data 2-18
Mapping List
Back To Add Button 2-8
Delete Button 2-8
Insert Button 2-8

Replace Button 2-8
Messaging Examples
Short Format 2 ..... **4-8**, 4-13
Messaging Formats
Short Format 2 ..... **1-6**
VisualMotion ASCII Format. **1-6**, **4-11**
most significant bit 3-6
Multiplex Data Bits ***4-1***
Multiplex Data Types . 1-3, 1-4, 2-11
Multiplexing
Control and Status Words **1-5**
Enable 2-14

N

nCyc Chan Ready 3-7
New
Cyclic, Multiplex Data 2-18
Cyclic, Single data 2-7
Non-Cyclic, Multiplex Data 2-19
Non-Cyclic, Single Data 2-9
Non-Cyc Ready 3-7
Non-Cyclic Channel 1-2, **1-7**
Non-Cyclic Data Channel
Multiplex Example 2-18
Non-Cyclic Data Mapping Lists
Direct Mapping
Basic Example 2-8
Multiplex Example **2-18**
Non-Cyclic Transmission
Data Exchange Objects 4-19
Number of Valid Data Bytes (#Bytes) **4-5**
nVM FW OK 3-7

O

OBJECT MAPPING LIST
Basic Example
Cyclic Data 2-6
Non-Cyclic Data 2-9
Multiplex Example
Cyclic Data 2-17
Non-Cyclic Data 2-18
OK, Send to CLC
Basic Example 2-5
Multiplex Example
Cyclic Data 2-15
output 1-2

P

Parameter Channel ... 1-2, 1-6, 2-20
Short Format 2 1-6
VisualMotion ASCII Format 1-6
Parameter Mode
Basic Example 2-7
Multiplex Example 2-18
Parameters 3-3
P-CHAN Components
Control/Status Word 4-4
Error Bit (E) **4-4**
first cycle **4-4**
last cycle **4-4**
Number of Valid Data Bytes (#Bytes) **4-5**
Toggle Bit (T) **4-5**
Transmission Format (Fmt) **4-5**
Pinouts (DPF05.x) 5-3
PLC Programming 4-1
Multiplex Data Bits ***4-1***
Non-Cyclic Data (via Data Exchange Objects) 4-19
Non-Cyclic Data (via Direct Mapping) 4-16
Parameter Channel 4-3
Print
File Menu entry 3-2
Print a Summary Report 3-2
Profibus DP Combi Slave Board . 5-1

R

Real-Time Channel 1-2, 1-3
Register 19 Definition (Fieldbus Status) 3-6
Register 20 Definition (Fieldbus Diagnostics) 3-7
Register 26 Definition (Fieldbus Resource Monitor) 3-8
Response Telegram Structure
Short Format 2 4-7
VisualMotion ASCII Format 4-12

S

Save to File
Cyclic, Multiplex Data 2-18
Cyclic, Single Data 2-7
Non-Cyclic, Multiplex Data 2-19
Non-Cyclic, Single Data 2-9
Send To CLC
Cyclic, Multiplex Data 2-18
Cyclic, Single Data 2-7
Non-Cyclic, Multiplex Data 2-19
Non-Cyclic, Single Data 2-9
Send to CLC
Basic Example
Cyclic Data 2-7
Non-Cyclic Data 2-9
Multiplex Example
Cyclic Data 2-17
Short Format 2 1-6
Command Telegram Structure 4-7
Messaging Example 4-13
Messaging Examples **4-8**
Response Telegram Structure 4-7
Shutdown CLC (Fieldbus Error Reaction) 3-10
Single Data Types 1-3, 1-4
Source Object list
Basic Example
Non-Cyclic Data 2-9
status word 1-5
Multiplex Example
Cyclic Data 2-14

T

To FieldBus
Basic Example 2-3
Multiplex Example
Cyclic Data 2-14
Toggle Bit (T) **4-5**
Transmission Format (Fmt) **4-5**

V

View a Summary Report 3-1
VisualMotion ASCII Format 1-6, **4-11**
Command Telegram Structure 4-11
Response Telegram Structure 4-12

W

Warning Only (Fieldbus Error Reaction) 3-10

X

X68 Connector.....5-3
X69 Connector.....5-3

engineering
mannesmann
Rexroth

Supplement B
DeviceNet Fieldbus Interfaces
VisualMotion 6.0

Indramat

Contents

1 General Information 1-1
1.1 CLC-D System Description with a Fieldbus 1-1
The VisualMotion Fieldbus Mapper 1-1
1.2 Data Transfer Direction (Output vs. Input) 1-2
1.3 Fieldbus Data Channel Descriptions 1-2
Cyclic Channel 1-2
The Real-Time Channel 1-3
Non-Cyclic Channel 1-6
Direct-Mapped Data 1-6
Data Exchange Objects 1-7

2 Fieldbus Mapper Examples 2-1
2.1 Basic Example 2-1
STEP I: Determine the Cyclic and Non-Cyclic Data 2-1
STEP II: Configure Fieldbus Slave Card with Setup Information and Cyclic Data Object Lists Using the Fieldbus Mapper 2-2
STEP III: Define Cyclic Data Mapping Lists 2-5
STEP IV: Define Non-Cyclic Data Mapping Lists (Direct Mapping) 2-8
2.2 Multiplexing Example 2-10
STEP I: Determining the Cyclic and Non-Cyclic Data (with Multiplexing) 2-10
STEP II: Configure Fieldbus Slave Card with Setup Information and Cyclic Data Object Lists Using the Fieldbus Mapper 2-11
STEP III: Define Cyclic Data Mapping Lists 2-15
STEP IV: Define Non-Cyclic Data Mapping Lists (Direct Mapping) 2-17

3 Information for the GPS Programmer 3-1
3.1 Viewing/Printing a Summary Report of the Current Fieldbus (only when CLC-D is on-line) 3-1
To View a Summary Report: 3-1
To Print a Summary Report: 3-2
3.2 Fieldbus-Accessible Parameters 3-3
3.3 Register 19 Definition (Fieldbus Status) 3-6
Diagnostic Object 5FF2 3-6
Bit Definitions 3-6
3.4 Register 20 Definition (Fieldbus Diagnostics) 3-7
Diagnostic Object 5FF0 3-7
Bit Definitions 3-8
3.5 Register 26 Definition (Fieldbus Resource Monitor) 3-8
Bit Definitions 3-9
3.6 Fieldbus Error Reaction 3-9

4 Information for the PLC Programmer 4-1
4.1 *.eds File 4-1
4.2 Word and Byte Swapping 4-1
Example: Allen-Bradley SDN Module for SLC-Series PLC 4-1
4.3 Multiplex Data Bits in the Control and Status Words 4-1
4.4 Non-Cyclic Transmission (Direct-Mapped Objects) 4-3
Selecting a Direct-Mapped Object 4-3
Transmission Sequence via a Direct-Mapped Object 4-3
Non-Cyclic Direct-Mapped Write 4-4
Non-Cyclic Direct-Mapped Read 4-5
4.5 Non-Cyclic Transmission (Data Exchange Objects) 4-6
Selecting a Data Exchange Object 4-6
Transmission Sequence via a Data Exchange Object 4-7

5 DeviceNet DCF01.x Slave Board 5-1
5.1 Application 5-1
5.2 Functionality 5-1
5.3 DCF01.x Card Hardware 5-2
Front View of the DCF01.x 5-2
DCF01.x Structure 5-2
Connector Pinouts 5-3
X78 Connector, DeviceNet 5-3
X77 Pinouts, External Inputs 5-3
DCF01.x Diagnostics 5-3
Front Panel LEDs 5-3
Definition of the Diagnostic LEDs 5-4
5.4 DeviceNet Polled I/O (Cyclic Channel) 5-5
DCF01.x Polled I/O Configuration 5-5
User-Specific Polled I/O Configuration 5-5
Monitoring the DCF01.x Polled I/O 5-5
Non-Cyclic Objects 5-5
Non-Cyclic, Data Exchange Objects 5-6
Non-Cyclic, Direct-Mapped Objects 5-6

6 Index 6-1

1 General Information

Important: Using a Fieldbus with the VisualMotion system will consume additional processing resources on the CLC-D card. Please take into consideration the consumption of these resources, especially when adding a Fieldbus to an existing application.

Version Note:
Information in this document is based on VisualMotion Toolkit software version 06V12 and CLC-D firmware version GPS06V64.

1.1 CLC-D System Description with a Fieldbus

The CLC-D can operate on a serial Fieldbus interface (network) by means of a plug-in card (DCF01.x board) that communicates with the CLC-D card via dual-port RAM. The function of the Fieldbus card is similar to that of a network card in a PC: it allows communication with other devices on the network.

In ***Figure 1-1: Sample Master/Slave Setup with Fieldbus Cards***, a commonly described Fieldbus interface is pictured:

- **Fieldbus Master** - PLC Fieldbus interface
- **Fieldbus Slave** - CLC-D Fieldbus interface

In this document, we will refer to the PLC as the **Fieldbus master** and the CLC-D as the **Fieldbus slave**.

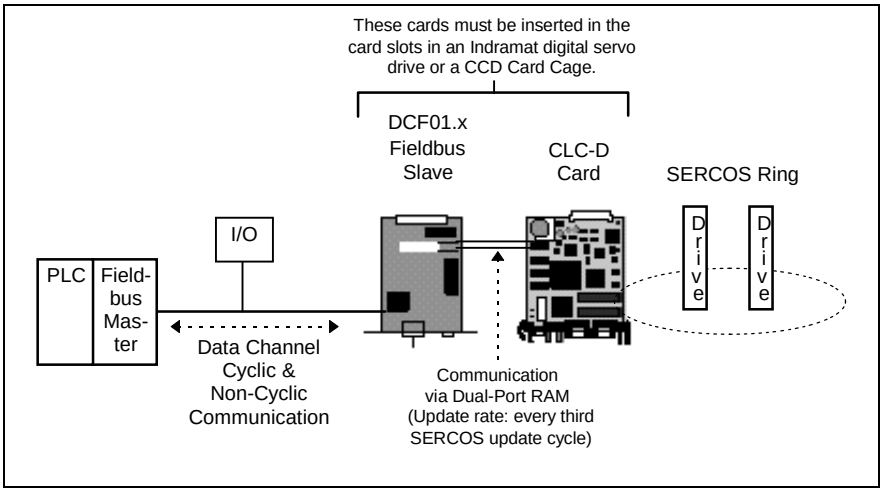

These cards must be inserted in the card slots in an Indramat digital servo drive or a CCD Card Cage.
DCF01.x Fieldbus Slave
CLC-D Card
SERCOS Ring
I/O
PLC
Field-bus Mas-ter
Drive
Drive
Data Channel Cyclic & Non-Cyclic Communication
Communication via Dual-Port RAM (Update rate: every third SERCOS update cycle)

Figure 1-1: Sample Master/Slave Setup with Fieldbus Cards

With the CLC-D card, the DeviceNet (DCF01.x) Fieldbus card can be used **only** as a **slave** (Group 2 only slave) card in a master/slave setup.

Important: When using a Fieldbus slave interface, it is recommended to use a 4 ms or higher SERCOS update.

The VisualMotion Fieldbus Mapper

In the VisualMotion software package, the Fieldbus Mapper is a tool used to set up Fieldbus data. It is primarily an on-line tool, used most effectively when connected to a CLC-D card.

Important: Whenever the Fieldbus Mapper is configured on-line, the system must be in Parameter Mode.

1.2 Data Transfer Direction (Output vs. Input)

In the VisualMotion Fieldbus Mapper, output and input are always described with respect to the Fieldbus master (the Fieldbus card associated with the PLC). The definitions for output and input follow:

output: the communication from the PLC to the CLC-D card (i.e. from the Fieldbus master to the Fieldbus slave).
Synonyms for this type of communication: **send** or **write** data.

input: the communication from the CLC-D card to the PLC (i.e. from the Fieldbus slave to the Fieldbus master).
Synonyms for this type of communication: **receive** or **read** data.

1.3 Fieldbus Data Channel Descriptions

The Indramat DeviceNet Fieldbus interface card for the CLC-D (DCF card) supports two data channels for input/output communication with the CLC-D:

1. **Cyclic Channel:** Polled I/O
   Real-Time Channel (for **single** and **multiplex** channels)
2. **Non-Cyclic Channel:** Explicit Messaging

Cyclic Channel

The DCF01.x Fieldbus card has pre-defined cyclic bus objects (16- or 32-bit) which are declared and transmitted as an ordered list (the bus configuration list). This bus configuration list acts as a placeholder for CLC-D mapped data from the CLC-D object mapping list. See *Object Lists and Their Transfer Locations.*

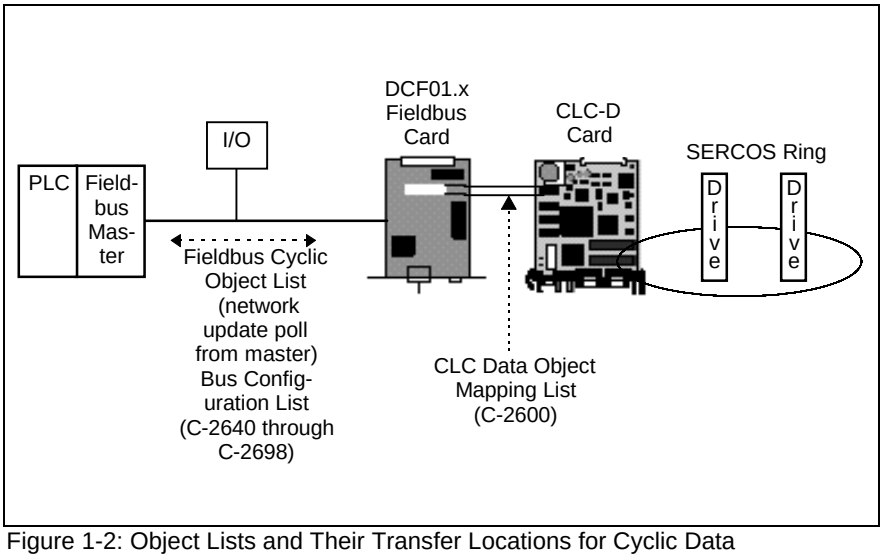


Figure 1-2: Object Lists and Their Transfer Locations for Cyclic Data

The cyclic data channel is limited to a total of 16 input words and 16 output words. CLC-D data types consume these objects in either one-word (or 16-bit) groups for CLC registers or two-word (or 32-bit) groups for all other data types in the object mapping list (C-0-2600).

Note: The bus configuration list must be set up in the Fieldbus mapper bus configuration screen before the object mapping list can be configured.

For the cyclic data, the CLC-D data object mapping list is scanned every third SERCOS update cycle and data is sent and received to/from the slave board's dual port RAM.

The cyclic data channel can be made up of any combination of the following data types:

- Real-Time Channel
  - Single Channel
  - Multiplex Channel

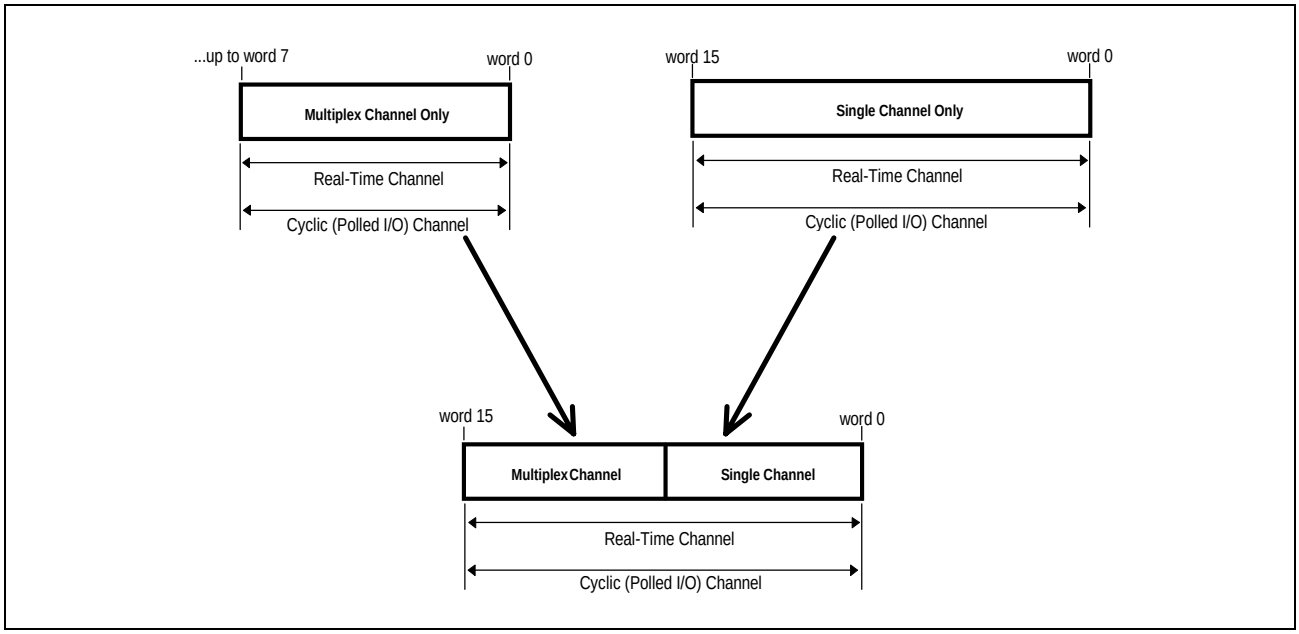


Figure 1-3: Configuration Options for the Cyclic Data Channel (3 possibilities)

The Real-Time Channel

In the real-time channel, data is updated cyclically between the Fieldbus master and slave. This channel contains two possible data types: **single** and **multiplex**.

Data Objects: Types and Sizes

The following table outlines the CLC-D data object types that can be transmitted via the cyclic channel and the amount of space (in 16-bit data words) that each data type consumes. Remember, the cyclic channel is limited to 16 data words in each direction (input and output).

Note: The cyclic data mapping list supports only 16- and 32-bit data of the following types for reading and writing:
- Integer
- Float
- Binary (used in CLC parameters)
- Hex (used in CLC parameters)

For all other data types (e.g. diagnostic messages - “strings”), use the non-cyclic Data Exchange Object or the Parameter Channel.

CLC Data Type	Data Size (in Data Words)
Register	1
Integer (currently active program ONLY *)	2
Float (currently active program ONLY *)	2
Global Integer	2
Global Float	2
CLC Card Parameter	2
CLC Axis Parameter	2
CLC Task Parameter	2
Note: Drive parameters "S" or "P" cannot be transmitted cyclically because of the inherent delay of parameter access over the SERCOS service channel. See ***"Non-Cyclic Channel/Data Exchange Objects"*** on page 1-8. However, if a drive parameter is mapped to an axis parameter, that axis parameter could be used in cyclic data (see description of Axis Parameters 180-196 in the **VisualMotion 6.0 Reference Manual**).	
*** Important Note:** Integers and floats are shown only for the currently active program. Each time you activate a new progam, the fieldbus reads/writes to the newly-activated program.	

Table 1-1: CLC-D Cyclic Data Types and Sizes

Single Data Types

Indramat Fieldbus interfaces have single (16-bit) and double (32-bit) word objects in the cyclic data channel with which simple applications can be handled without any difficulty. These objects are directly mapped to CLC-D data types and updated every third SERCOS update cycle.

Multiplex Data Types (Cyclic Data Channel)

In some multi-axis applications, 16 words of cyclic data transfer are not sufficient to meet the data transfer requirement of the application.

When insufficient data transfer space is available, multiplex data can be set up within the cyclic channel. One data object (base object) acts as a placeholder for multiple possible CLC-D data types (all of the same word size). The currently transmitted CLC-D data type is based on an index value placed in a multiplex control word attached to the end of the cyclic data list. Depending on the index specified by the master, the multiplex channel permits a different set of data within the cyclic channel to be transferred as current real-time data in both data directions.

Note: Using the multiplex channel will reduce the maximum number of usable words for storing CLC data to 15. The 16th word (or last used word, if fewer than 15 words) is used as the multiplex entry control/status word.

In the Fieldbus configuration lists, the following multiplex objects are available:

Type of Multiplex Object	Input Objects Available	Total Input Objects	Output Objects Available	Total Output Objects
32-bit	3 base (x16 indices)	48	3 base (x16 indices)	48
16-bit	2 base (x16 indices)	32	2 base (x16 indices)	32

Table 1-2: Available Multiplex Objects

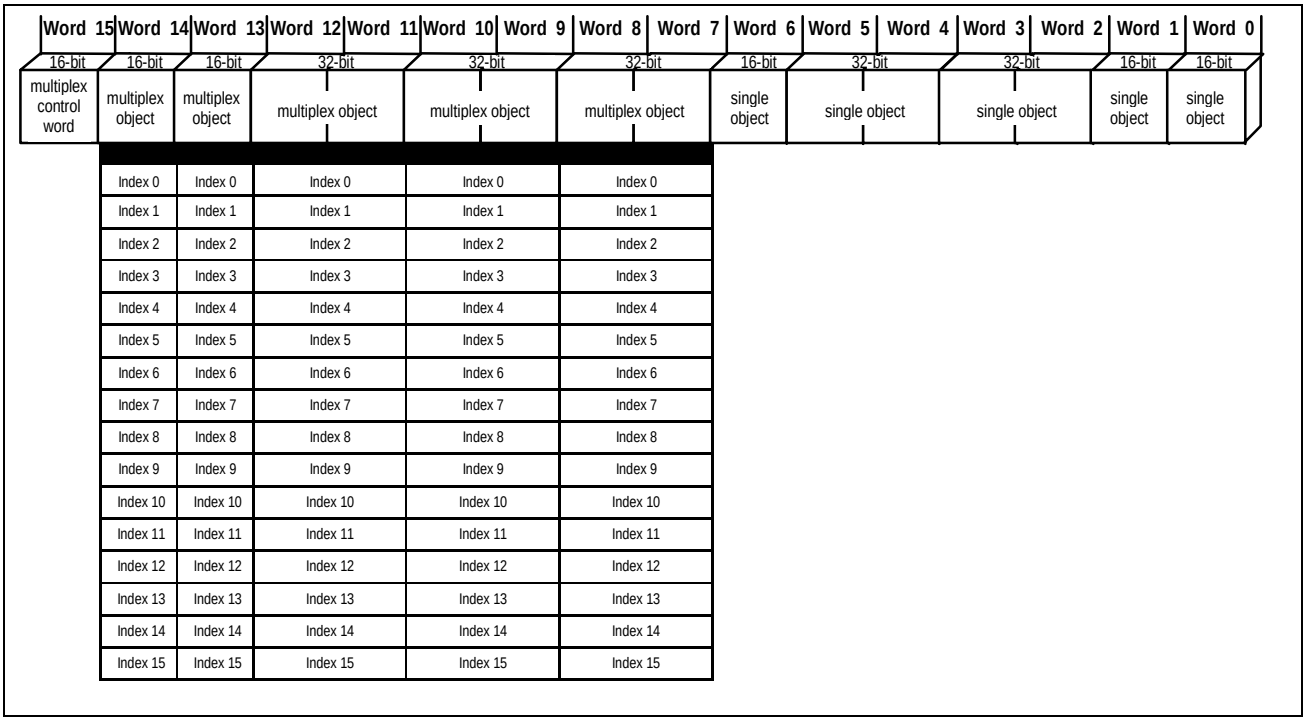

Word 15 Word 14 Word 13 Word 12 Word 11 Word 10 Word 9 Word 8 Word 7 Word 6 Word 5 Word 4 Word 3 Word 2 Word 1 Word 0
16-bit 16-bit 16-bit 32-bit 32-bit 32-bit 16-bit 32-bit 32-bit 16-bit 16-bit
multiplex control word
multiplex object
multiplex object
multiplex object
multiplex object
multiplex object
single object
single object
single object
single object
single object
Index 0 Index 0 Index 0 Index 0 Index 0
Index 1 Index 1 Index 1 Index 1 Index 1
Index 2 Index 2 Index 2 Index 2 Index 2
Index 3 Index 3 Index 3 Index 3 Index 3
Index 4 Index 4 Index 4 Index 4 Index 4
Index 5 Index 5 Index 5 Index 5 Index 5
Index 6 Index 6 Index 6 Index 6 Index 6
Index 7 Index 7 Index 7 Index 7 Index 7
Index 8 Index 8 Index 8 Index 8 Index 8
Index 9 Index 9 Index 9 Index 9 Index 9
Index 10 Index 10 Index 10 Index 10 Index 10
Index 11 Index 11 Index 11 Index 11 Index 11
Index 12 Index 12 Index 12 Index 12 Index 12
Index 13 Index 13 Index 13 Index 13 Index 13
Index 14 Index 14 Index 14 Index 14 Index 14
Index 15 Index 15 Index 15 Index 15 Index 15

Figure 1-4: Sample Command (Write) to Slave (PLC→CLC)

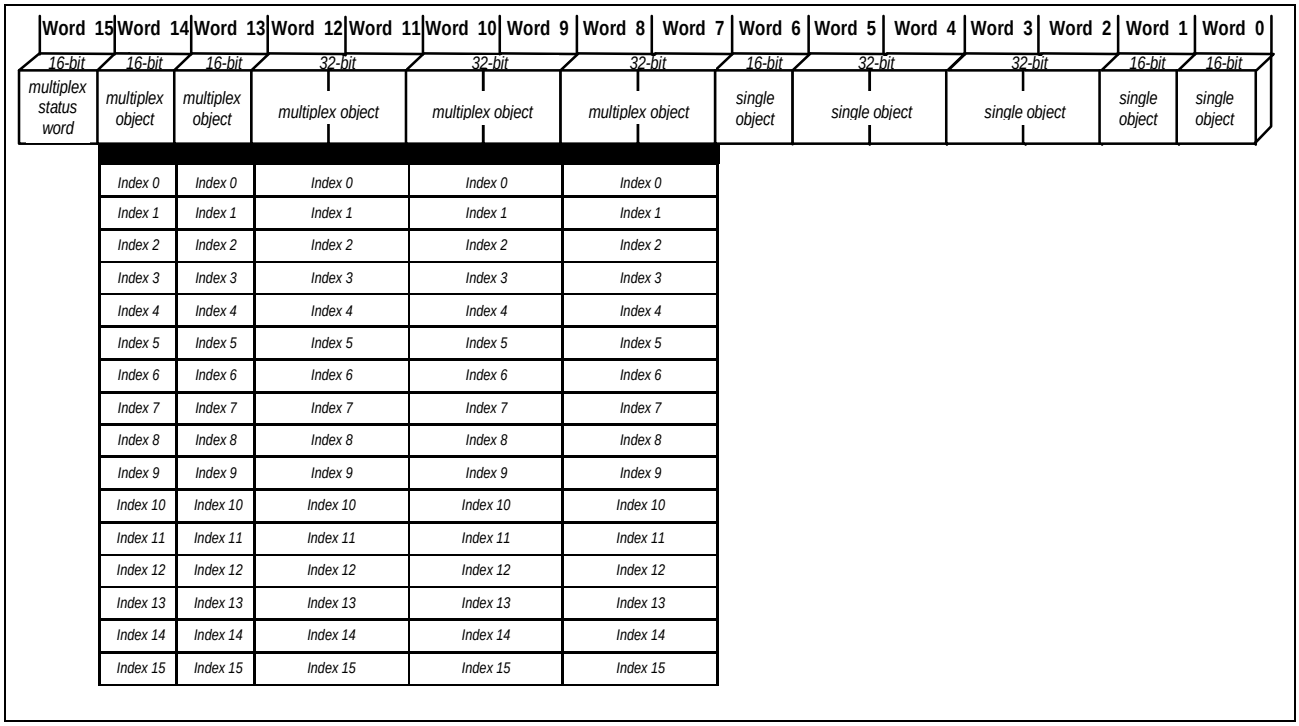

Word 15 Word 14 Word 13 Word 12 Word 11 Word 10 Word 9 Word 8 Word 7 Word 6 Word 5 Word 4 Word 3 Word 2 Word 1 Word 0
16-bit 16-bit 16-bit 32-bit 32-bit 32-bit 16-bit 32-bit 32-bit 16-bit 16-bit
multiplex status word
multiplex object
multiplex object
multiplex object
multiplex object
multiplex object
single object
single object
single object
single object
single object
Index 0 Index 0 Index 0 Index 0 Index 0
Index 1 Index 1 Index 1 Index 1 Index 1
Index 2 Index 2 Index 2 Index 2 Index 2
Index 3 Index 3 Index 3 Index 3 Index 3
Index 4 Index 4 Index 4 Index 4 Index 4
Index 5 Index 5 Index 5 Index 5 Index 5
Index 6 Index 6 Index 6 Index 6 Index 6
Index 7 Index 7 Index 7 Index 7 Index 7
Index 8 Index 8 Index 8 Index 8 Index 8
Index 9 Index 9 Index 9 Index 9 Index 9
Index 10 Index 10 Index 10 Index 10 Index 10
Index 11 Index 11 Index 11 Index 11 Index 11
Index 12 Index 12 Index 12 Index 12 Index 12
Index 13 Index 13 Index 13 Index 13 Index 13
Index 14 Index 14 Index 14 Index 14 Index 14
Index 15 Index 15 Index 15 Index 15 Index 15

Figure 1-5: Sample Response (Read) to Master (CLC→PLC)

Multiplex Control and Status Words

The DeviceNet multiplex control and status words serve to command and acknowledge multiplex data transferred between the Fieldbus master and the Fieldbus slave. The **control** word is associated with **output** communication (PLC→CLC). The **status** word is associated with **input** communication (CLC→PLC). Single data objects are not affected by the multiplex control and status words.

Note: For specific information about how the Fieldbus master uses the multiplex control and status words, see ***Multiplex Data Bits in the Control and Status Words*** on page 4-1.

Sending and Receiving the Same CLC-D Data Cyclically

Two separate objects will be consumed when sending and receiving the same CLC-D data (both cyclically and non-cyclically).

If you wish to cyclically send and receive the same CLC-D data, the Output mappings should come **first** in the list (see ***Fieldbus Mapper Examples, Basic Example*** on page 2-1).

Non-Cyclic Channel

The non-cyclic channel is used for data that needs to be transferred only once or sporadically, such as:

- the transmission of lists
- parametriziation of axes or programs
- any non-cyclically mapped data

Instead of being updated during each cycle, non-cyclic data is transferred whenever time is available on the Fieldbus. Though any data type can be transferred non-cyclically, diagnostic messages and drive parameters (S and P) **must** be transferred non-cyclically because of the non-cyclic retrieval for drive parameters through SERCOS and the length of the diagnostic messages.

For example: ASCII text in a diagnostic message requires one data word for every two ASCII characters. The non-cyclic channel can transfer only up to 16 data words (or 32 characters) of a diagnostic message at once. This would mean that one diagnostic text message (if 32 characters or more) could consume all the available cyclic data. For this reason, the transfer of diagnostic messages must be made over the non-cyclic (Explicit Messaging) or Parameter Channel; it is not allowed over the real-time channel.

There are two types of non-cyclic data transmissions for the CLC-D/VisualMotion system:

- data transmitted via the data exchange object
- data mapped directly to CLC-D data types

Non-cyclic data can be accessed via Explicit Messaging support of the Fieldbus master

Direct-Mapped Data

Just as there are pre-defined cyclic data objects for mapping CLC data to the DeviceNet Polled I/O channel, there are also pre-defined non-cyclic data objects (16- and 32-bit) that can be mapped to CLC data types.

Data is mapped via the non-cyclic data mapping list (C-0-2700). It provides a means for the master to easily access non-cyclic data. However, the fixed mapping list is limited to a certain number and types of CLC-D data.
The directly-mapped non-cyclic data (take note of size and direction) can be assigned to the following objects:

Number of Objects Available	Data Size	Direction (reference: Master Fieldbus)	DeviceNet Names of Objects	Indramat Object ID
16	32-bit	in	Class 110 Instance 1-16 Attribute 100	5F10-5F1F
16	32-bit	out	Class 109 Instance 1-16 Attribute 100	5F00-5F0F
32	16-bit	in	Class 115, 119 Instance 1-16 Attribute 100	5F60-5F6F, 5FA0-5FAF
32	16-bit	out	Class 116, 120 Instance 1-16 Attribute 100	5F70-5F7F, 5FB0-5FBF

Table 1-3: Non-Cyclic (Direct-Mapped) DeviceNet Data Objects

Note: The available CLC data types for directly-mapped non-cyclic data are the same as for directly-mapped cyclic data.

Data Object Types and Sizes

The following table outlines the directly-mapped CLC-D data object types that can be transmitted via the non-cyclic channel and the amount of space (in data words) that each data type consumes.

CLC Data Object Type	Data Size (in Data Words)
Register	1
Integer (currently active program ONLY)	2
Float (currently active program ONLY)	2
Global Integer	2
Global Float	2
CLC Card Parameter	2
CLC Axis Parameter	2
CLC Task Parameter	2

Table 1-4: Data Object Types/Sizes

Sending and Receiving the Same CLC-D Data Non-Cyclically

Two separate objects will be consumed when sending and receiving the same CLC-D data (both cyclically and non-cyclically).

Data Exchange Objects

Four data exchange objects Class 100, Instance 1-4, Attribute 100 are available for the transfer of non-cyclic data. These objects represent fixed data "containers" of varying lengths that transfer the VisualMotion ASCII Protocol to the CLC-D card, in the same way that data is transferred using the VisualMotion ASCII Format via an Explicit Message. These objects serve as an open-ended possibility to access any VisualMotion data (including cams, diagnostic text, etc.), but more work is required in the master to perform a transmission of this type. For more specific information about these objects, see ***Non-Cyclic Transmission (Data Exchange Objects)*** on page 4-6.

2 Fieldbus Mapper Examples

2.1 Basic Example

The following example demonstrates the process for configuring the mapping of basic data between the CLC-D card and a DeviceNet Fieldbus.

Important: The Fieldbus Mapper is an on-line configuration tool. The following conditions must be met when performing STEP I through STEP IV below:
- The system must be powered up.
- The CLC-D card must be connected to the system.
- The system must be set to parameter mode.

STEP I: Determine the Cyclic and Non-Cyclic Data

Cyclic Data

In this example, we will assume that you want to cyclically transfer the following data between the CLC-D card and the Fieldbus master in the order listed.

Note: Quantity and type of data varies depending on application.

Output Data (From Bus to CLC Card)	Size (Data Words)	Size (Bits)	Input Data (From CLC Card to Bus)	Size (Data Words)	Size (Bits)
Register 100	1	16	Integer 29	2	32
Float 2	2	32	Register 47	1	16
Global Integer 3	2	32	Register 120	1	16
Register 104	1	16	Global Float 22	2	32
Register 40	1	16	Axis Parameter 1.100	2	32
Integer 4	2	32	Integer 30	2	32
Integer 5	2	32			
TOTAL	11 Data Words			10 Data Words	

Table 2-1: Sample Cyclic Data

Non-Cyclic Data

We will assume that the following data will be transferred non-cyclically for system initialization or setup. Remember, the direct-mapped non-cyclic data is intended for infrequent data transfer.

Output Data (From Bus to CLC Card)	Size (Data Words)	Size (Bits)	Input Data (From CLC Card to Bus)	Size (Data Words)	Size (Bits)
Float 16	2	32	Float 5	2	32
Integer 7	2	32	Integer 8	2	32
Register 101	1	16			

Table 2-2: Sample Non-Cyclic Data

Note: See ***Table 1-3: Non-Cyclic (Direct-Mapped) DeviceNet*** Data Objects for data limitations.

Note: For other data types, such as diagnostic text messages, or S and P parameters, use the data exchange object (see ***Non-Cyclic Channel*** on page ***1-6***).

STEP II: Configure Fieldbus Slave Card with Setup Information and Cyclic Data Object Lists Using the Fieldbus Mapper

Important: The drive must be set to Parameter Mode to configure and assign the data mapping for Fieldbuses (STEP II and STEP III).

1. In the VisualMotion "Data" Menu, select "Fieldbus Mapper."

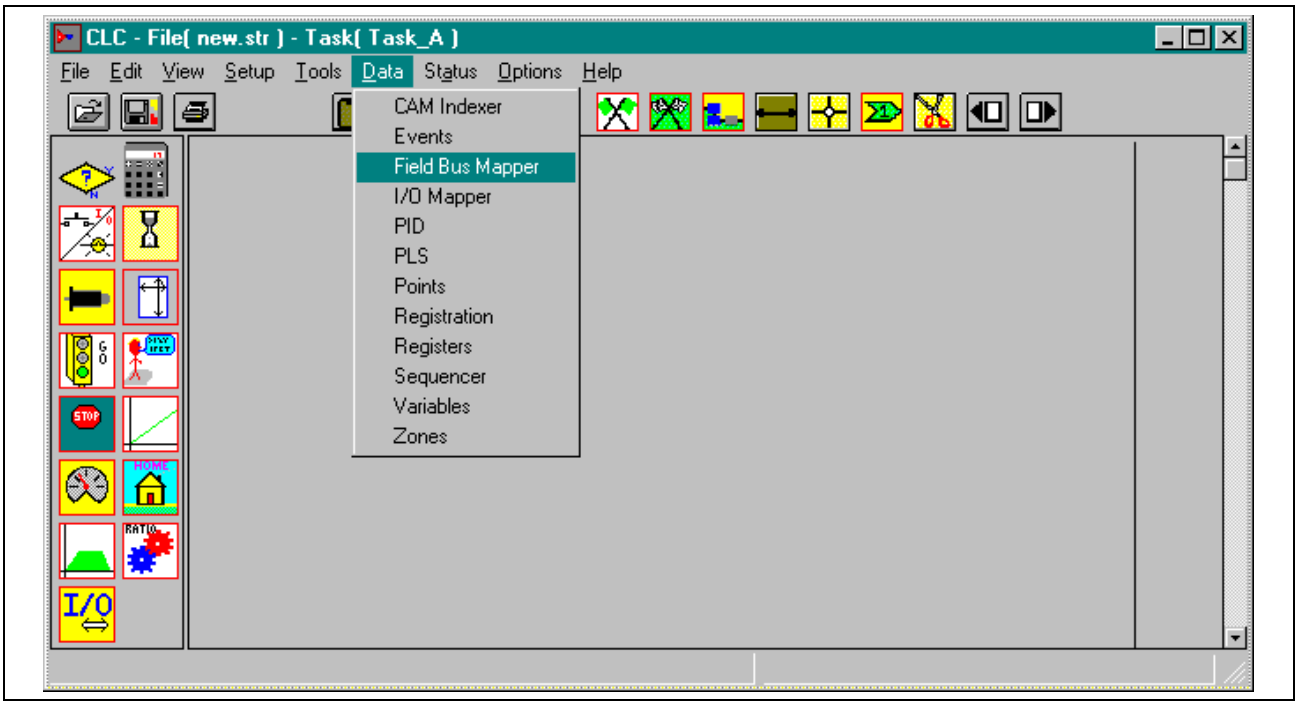


Figure 2-1: Selecting the Fieldbus Mapper from the main VisualMotion Screen

2. In the scroll box under "Bus Type," ensure that the desired bus is selected (If the DCF card is connected, DeviceNet should appear).
3. Choose the cyclic data channel (Polled I/O).

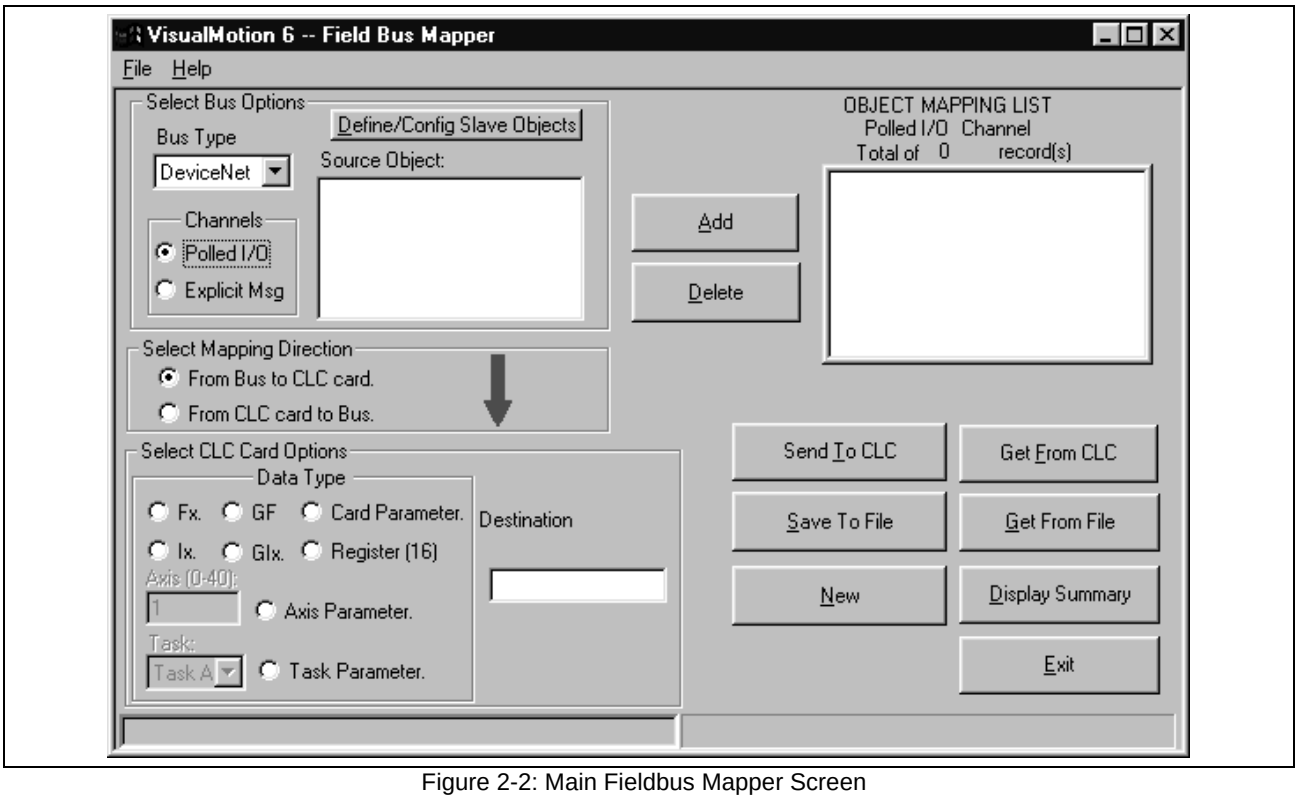


Figure 2-2: Main Fieldbus Mapper Screen

4. Click the button labeled "Define/Config Slave Objects." The Fieldbus Mapper will try to detect any configuration data from the CLC-D. If you are configuring a new list, click on the "NEW" button to clear any current data in the selected list. The DeviceNet Configuration Screen is pictured in ***Figure 2-3: DeviceNet Configuration Screen***.

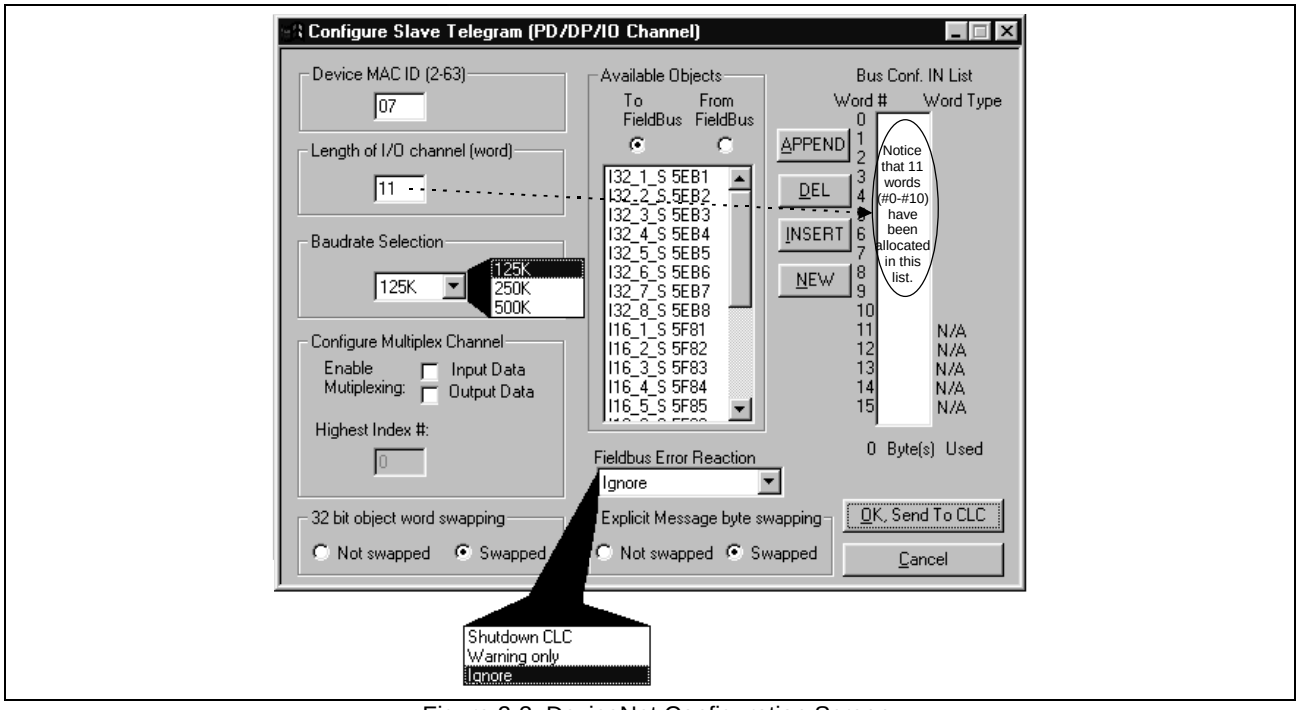


Figure 2-3: DeviceNet Configuration Screen

5. Set the device MAC ID address to a unique number for this device on the bus (2-63).
6. Set the "Length of I/O Channel (Word)" to the number of data words to be allocated for the input channel or the output channel, whichever is greater. According to the example data in ***Table 2-1: Sample Cyclic Data*** under "STEP I: Determine the Cyclic and Non-Cyclic Data," we have set the length of the channel to 11 for the output channel (made up of [3] 16-bit objects of 1 data word each and [4] 32-bit objects of 2 data words each). The input channel contains only 10 words, so a “filler” object must be placed in the Bus Configuration IN list for the 11th word.
7. Select the baud rate in the "Baudrate" combo box to match that of the other network devices.
8. Select "Not swapped" (default) or "Swapped" under "32 bit object word swapping," as required by your PLC. See ***Word and Byte Swapping*** on page 4-1.
9. Select "Not swapped" (default) or "Swapped" under "Explicit message byte swapping," as required by your PLC. See ***Word and Byte Swapping*** on page 4-1.
10. Ensure that the combo box under "Fieldbus Error Reaction" is currently set to "Ignore." Any other setting during setup will cause repeated error messages. Set the desired Error Reaction when the machine is ready for commissioning. See **Fieldbus Error Reaction** on page 3-9 for detailed information about the possible settings.
11. Choose the radio button below the words "To FieldBus."
12. For each data type desired, choose a corresponding available object in the list on the left and click "APPEND" to place it in the "Bus Conf. IN List." If you are configuring a new list, click on the "NEW" button to clear any current data in the selected list. Each available object has a typecode to identify its size and type. ***Figure 2-4: Configuring the DeviceNet Configuration IN List*** contains a description of the typecode.

Note: The desired order of the data on the master determines the object types (16- or 32-bit) and order of objects you place in the bus configuration list.

Note: Because the number of data objects in the IN list is less than the number in the OUT list, the remaining data word in the IN list must be assigned an available object that will not be used.

Following are descriptions of the buttons to manipulate the bus configuration lists:

inserts an available object after the last word placed in the current list.

removes the selected object from the current list.

inserts an available object above the selected object in the current list.

clears up the current list (only in the direction selected under “Available Objects”).

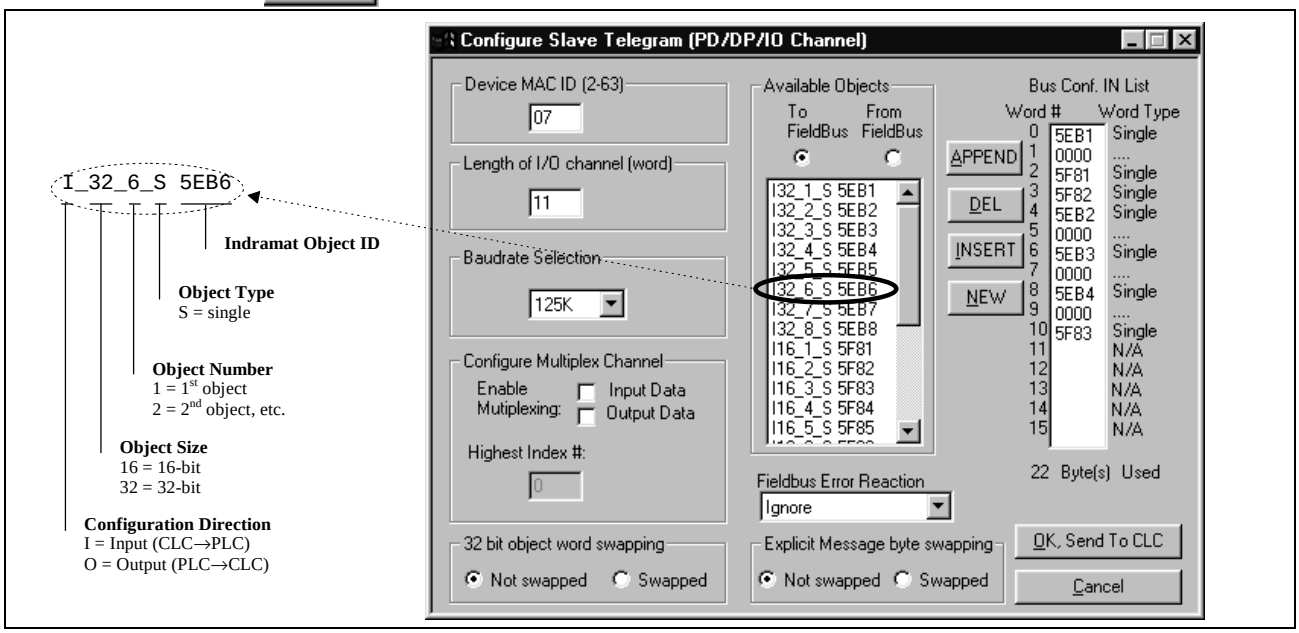


Figure 2-4: Configuring the DeviceNet Configuration IN List

12. Choose the radio button below the words "From FieldBus."

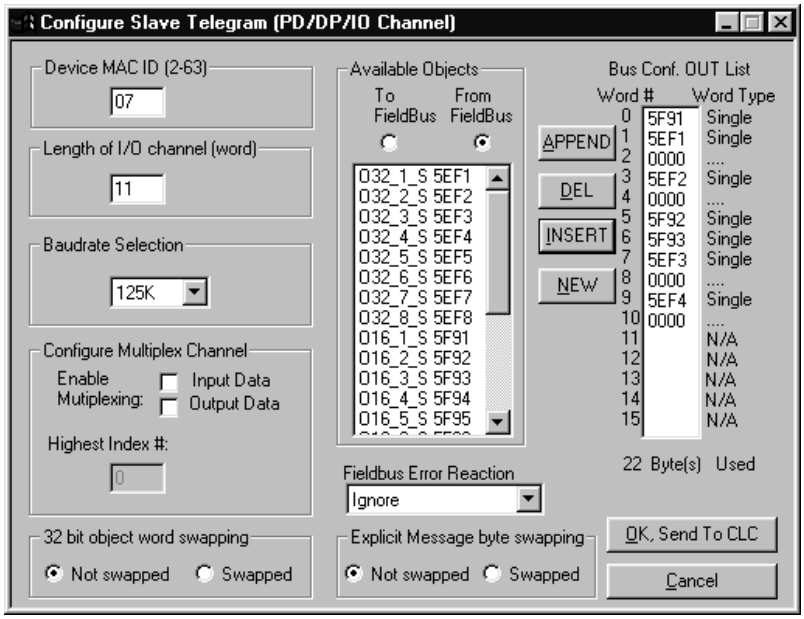


Figure 2-5: Configuring the DeviceNet Configuration OUT List

13. For each data type desired, choose a corresponding available object in the list on the left and click "APPEND" to place it in the "Bus Conf. OUT List." If you are configuring a new list, click on the "NEW" button to clear any current data in the selected list.

    We have now allocated the data objects in the cyclic channel so the desired data can be transferred between the CLC-D card and the DeviceNet Fieldbus card (see ***Figure 2-5: Configuring the DeviceNet Configuration OUT List***).

Note: The non-cyclic data requires no configuration of objects in a list. These objects are addressed directly and preassigned as non-cyclic. They need only to be mapped to the CLC-D data.

14. Click "OK, Send to CLC." The following window appears:

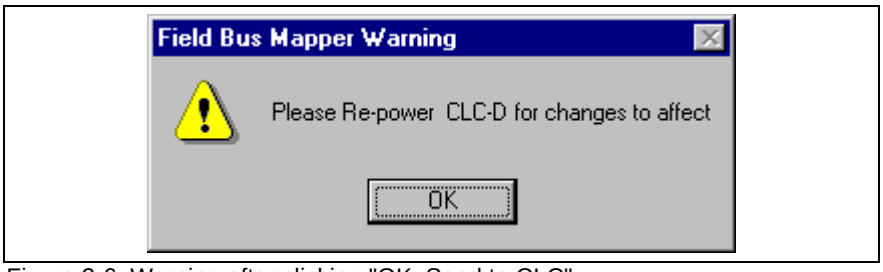


Figure 2-6: Warning after clicking "OK, Send to CLC"

STEP III: Define Cyclic Data Mapping Lists

STEP III associates the CLC-D data types to the objects assigned in STEP II.

In our example, the objects to be added are:

Output Data (From Bus to CLC Card)	Assigned Object	Input Data (From CLC Card to Bus)	Assigned Object
Register 100	5F91	Integer 29	5EB1
Float 2	5EF1	Register 47	5F81
Global Integer 3	5EF2	Register 120	5F82
Register 104	5F92	Global Float 22	5EB2
Register 40	5F93	Axis Parameter 1.100	5EB3
Integer 4	5EF3	Integer 30	5F83
Integer 5	5EF4		

Adding Objects to the Cyclic Data Mapping List

1. Ensure that the designated bus type is correct (DeviceNet).
2. Choose the desired cyclic data channel (Polled I/O).
3. Select the desired Mapping Direction.
   Our example begins with "From Bus to CLC Card."
4. Select the Data Type.
   In our example, the first item to be added to this list is Register 100.
   Select "Register" as the data type.

Note: The CLC-D will show only the objects associated with the chosen data type in the Source Object list. For example, if you have selected the mapping direction "From Bus to CLC Card" and designate "Register" as the data type, only 16-bit output objects that have been mapped in the bus configuration list (see STEP II) will be shown in the Source Object list.

5. Select or fill in the Destination. In our example, we must select Register 100 from the list in the pop-up screen shown in ***Figure 2-7: Sample Selection List*** OR type 100 as the destination.

Note: The Fieldbus Mapper selection lists display only the labels of the currently active program.

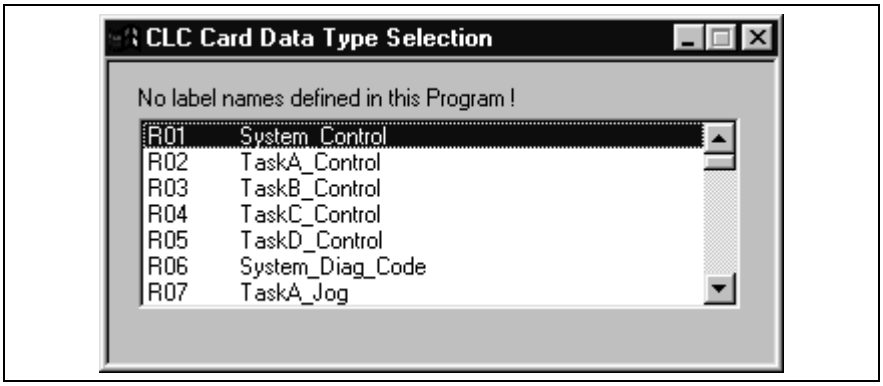


Figure 2-7: Sample Selection List

6. The source objects that are applicable to this data type will appear in the Source Object list. Choose the first available object.
7. Click the "Add" button. The object will appear in the "OBJECT MAPPING LIST" at the right.

Note: This step associates the object assigned in STEP II to the CLC-D data. The CLC-D scans the list created in this step and executes each mapping instruction **in order** during every third SERCOS update cycle.

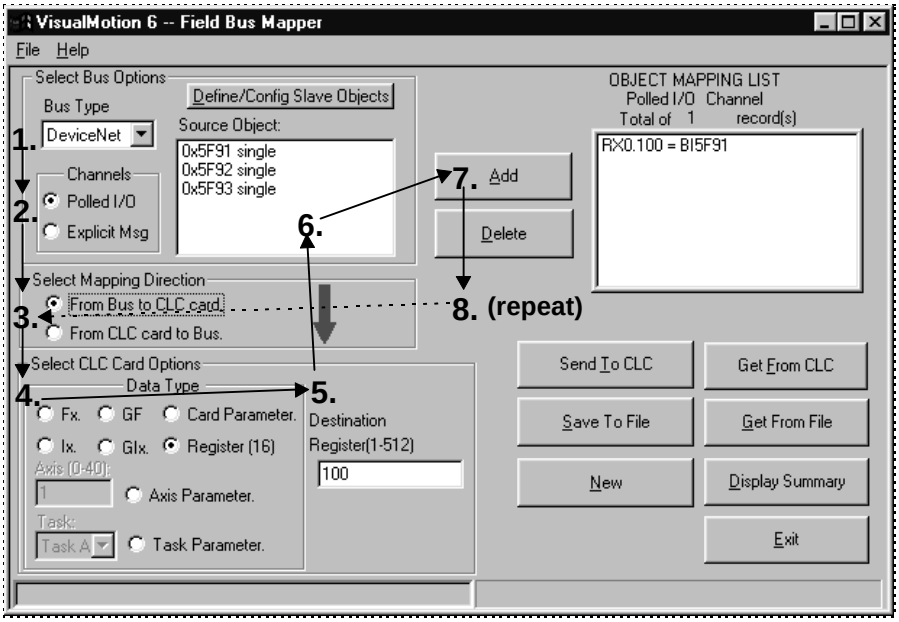


Figure 2-8: Adding Objects to the Cyclic Object Mapping List

Important: Cyclic data uses the objects configured in STEP II.

8. Repeat steps 3 through 7 above until each of the desired cyclic data objects has been added to the OBJECT MAPPING LIST.

Note: When mapping a PLC output object to a parameter, the parameter must be of the type "Read/Write Any Time." For parameters that are "Read Any Time/Write in Parameter Mode" or "Read Only," an error message is generated when you click “Send to CLC” if you try to map the parameters to a PLC output object. Examples: C-0-1550 (Read Any Time / Write in Parameter Mode) and C-0-1518 (Read Only).

9. Click “Send to CLC” to save the selected object mapping list to the CLC-D card. Use any of the following buttons, as needed.

Send To CLC	Saves the currently selected object mapping list to the CLC-D Card (to card parameter C-0-2600 if the cyclic [Polled I/O] channel is selected, and C-0-2700 if the non-cyclic [Explicit Messaging] channel is selected).
Save To File	Saves the currently selected object mapping list to a file (with a .prm extension).
New	Clears up the current object mapping list.
Get From CLC	Gets the currently selected object mapping list from the CLC-D Card (from card parameter C-0-2600 if the cyclic [Polled I/O] channel is selected, and C-0-2700 if the non-cyclic [Explicit Messaging] channel is selected).
Get From File	Allows the user to open a previously saved file (with a .prm extension) of an object mapping list. When the file is opened, displays that object mapping list.
Display Summary	Generates an on-line summary report of the currently selected Fieldbus. This report can be printed by selecting "Print" from the File menu in the Fieldbus Mapper window.
Exit	Exits the Fieldbus Mapper utility.

Important: Ensure that your system is set to Parameter Mode for the changes to take place.

Note: For debugging purposes, the Fieldbus Mapper follows each mapped item with the cursor as it is saved to the CLC-D. If there is a problem with the mapping of a particular object, an error message appears while the cursor remains on the object.

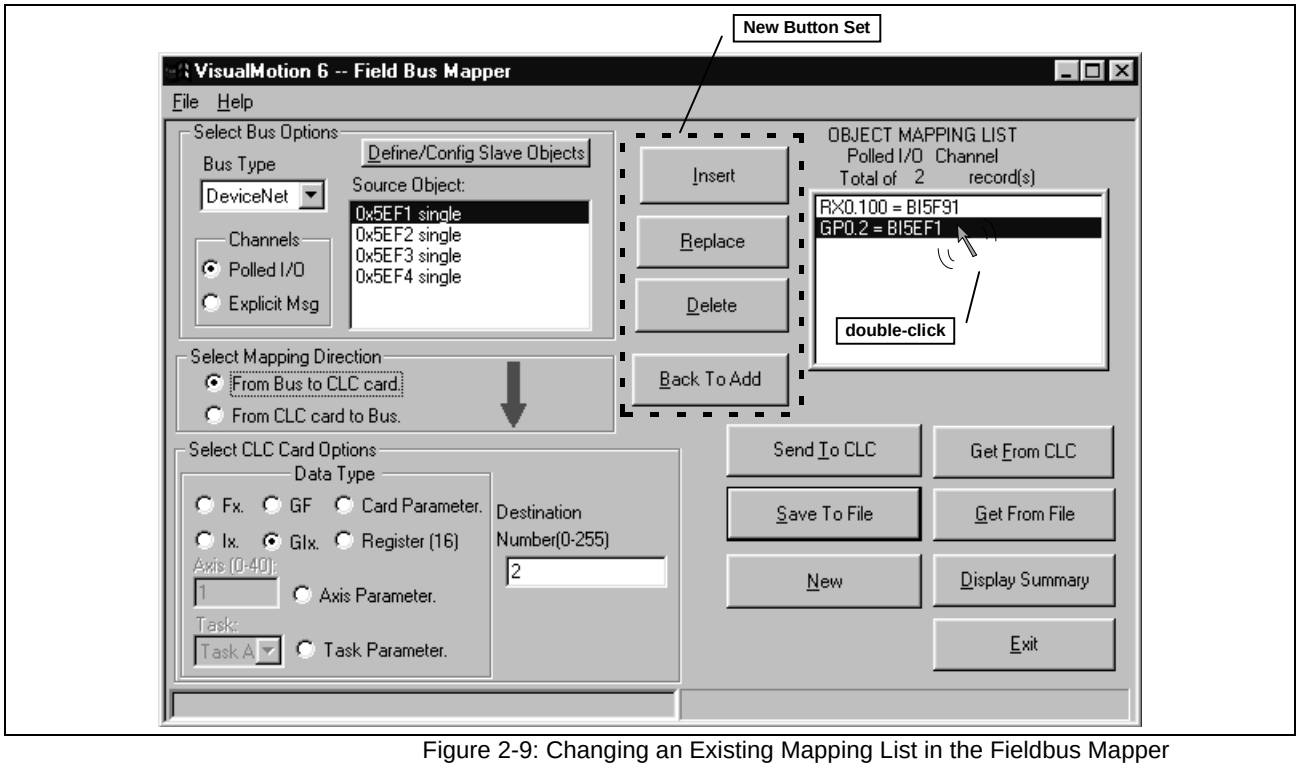


Figure 2-9: Changing an Existing Mapping List in the Fieldbus Mapper

Changing an Existing Object Mapping List

If you want to insert, replace or delete items in the Object Mapping List, double-click on the item in question. A new set of buttons appears to the left of the Object Mapping List box.

Inserts a new object into the list directly before the selected object.

Replaces the selected object with a new object.

Removes the selected object from the list.

Returns to Fieldbus mapper normal mode, which allows adding new items to the end of the list and deleting items.

STEP IV: Define Non-Cyclic Data Mapping Lists (Direct Mapping)

The non-cyclic data uses objects that are pre-defined for this type of data.

In our example, the following data will be transmitted non-cyclically:

Output Data (From Bus to CLC Card)	Input Data (From CLC Card to Bus)
Float 16	Float 5
Integer 7	Integer 8
Register 101	

Table 2-3: Objects to be transferred non-cyclically

Adding Objects to the Non-Cyclic Data Mapping List (see *Figure 2-10: Adding Objects to the Non-Cyclic Object Mapping List*)

1. Ensure that the designated bus type is correct.
2. Choose the non-cyclic data channel (Explicit Messaging).
3. Select the desired Mapping Direction.
   In our example, we will begin with the output list, or "From Bus to CLC."
4. Select the Data Type.
   In our example, the first item to be added to this list is Float 16. Select "Float" as the data type.
5. Fill in the Destination.
   In our example, we must type 16 as the destination.
6. The source objects that are applicable to this data type will appear in the Source Object list.
   Choose the first available object.
7. Click the "Add" button.
   The object will appear in the "OBJECT MAPPING LIST" at the right. This list is scanned by the CLC only when a non-cyclic request comes in and executes the mapping.
8. Repeat steps 2 through 7 above until each of the desired non-cyclic objects has been added to the OBJECT MAPPING LIST in both directions ("From Bus to CLC card" AND "From CLC card to Bus").

Note: If you want to insert, replace or delete items in the Object Mapping List, double-click on the item in question. A new set of buttons appears to the left of the Object Mapping List box. See ***Changing an Existing Object Mapping List*** above for a detailed explanation of each button.

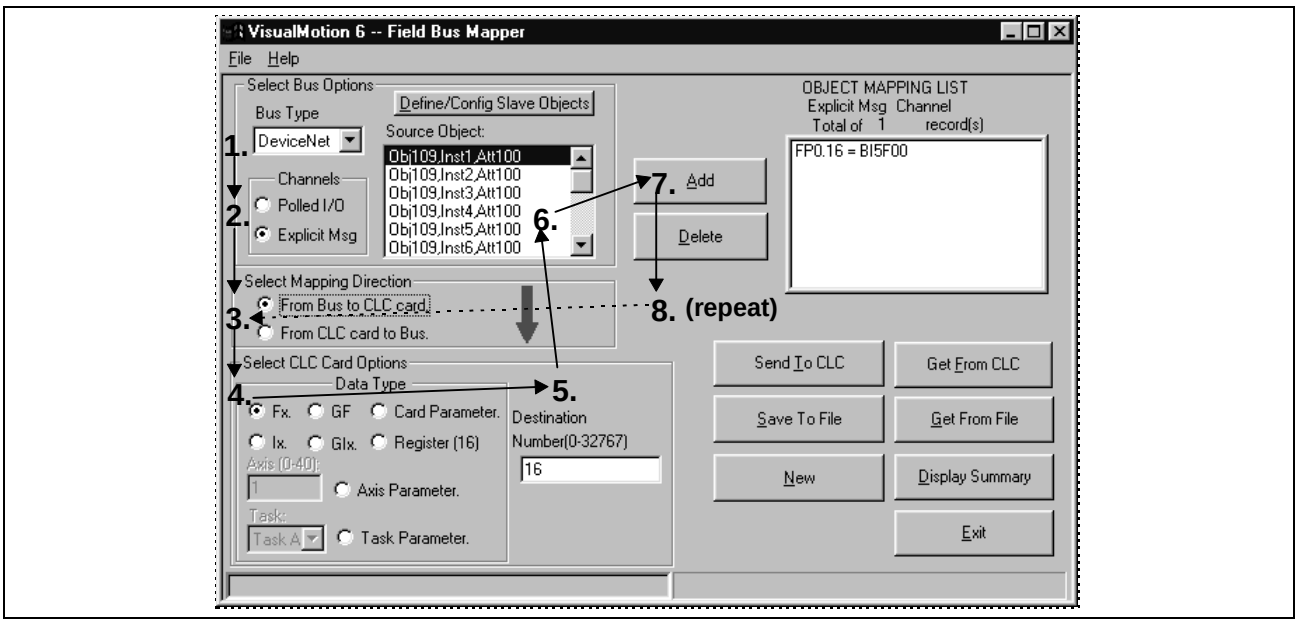


Figure 2-10: Adding Objects to the Non-Cyclic Object Mapping List

9. Click “Send to CLC” to save the selected object mapping list to the CLC-D card. Use any of the following buttons, as needed.

Saves the currently selected object mapping list to the CLC-D Card (to card parameter C-0-2600 if the cyclic [Polled I/O] channel is selected, and C-0-2700 if the non-cyclic [Explicit Messaging] channel is selected).

Saves the currently selected object mapping list to a file (with a .prm extension).

Clears up the current object mapping list.

Gets the currently selected object mapping list from the CLC-D Card (from card parameter C-0-2600 if the cyclic [Polled I/O] channel is selected, and C-0-2700 if the non-cyclic [Explicit Messaging] channel is selected).

Allows the user to open a previously saved file (with a .prm extension) of an object mapping list. When the file is opened, displays that object mapping list.

Generates an on-line summary report of the currently selected Fieldbus. This report can be printed by selecting "Print" from the File menu in the Fieldbus Mapper window.

Exits the Fieldbus Mapper utility.

Note: For debugging purposes, the Fieldbus Mapper follows each mapped item with the cursor as it is saved to the CLC-D. If there is a problem with the mapping of a particular object, an error message appears while the cursor remains on the object.

10. **Mapping is complete!**
For programming information, refer to ***Information for the GPS Programmer*** on page 3-1or ***Information for the PLC Programmer*** on page 4-1.

2.2 Multiplexing Example

Multiplexing is available in the cyclic (Polled I/O) channel for applications where more than the allotted 16 data transfer words are required. For a more detailed description of multiplexing, see ***Multiplex Data Types*** (Cyclic Data Channel) on page 1-4.

Important: The Fieldbus Mapper is an on-line configuration tool. The following conditions must be met when performing STEP I through STEP IV below:
- The system must be powered up.
- The CLC-D card must be connected to the system.
- The system must be set to parameter mode.

STEP I: Determining the Cyclic and Non-Cyclic Data (with Multiplexing)

Cyclic Data

A typical multiplexing example in a multi-axis application is to assign cyclic data to each index by axis (e.g. index 0 to axis 1, index 1 to axis 2, etc.). Although this example shows data for only three axes in this manner, remember that multiplexing allows sending up to 16 unique pieces of data for each multiplex object. Single and multiplex objects can be combined to fill up the first 15 words of the list.

In this example, we will assume that you want to cyclically transfer the following data between the CLC-D card and the Fieldbus master in the order listed:

Object Type, Size (bits)	Output Data (From Fieldbus Master to CLC Card)			Size (Data Words)	Object Type, Size (bits)	Input Data (From CLC Card to Fieldbus Master)			Size (Data Words)
	Axis 1 (Index 0)	Axis 2 (Index 1)	Axis 3 (Index 2)			Axis 1 (Index 0)	Axis 2 (Index 1)	Axis 3 (Index 2)	
Single, 16-bit	Register 100			1	Single, 16-bit	Register 120			1
Single, 32-bit	Float 2			2	Single, 32-bit	Global Float 22			2
Single, 32-bit	Integer 4			2	Single, 32-bit	Global Integer 44			2
Multiplex, 16-bit	Control Registers (preassigned per axis)				Multiplex, 16-bit	Status Registers (preassigned per axis)			
	Register 11	Register 12	Register 13	1		Register 31	Register 32	Register 33	1
Multiplex, 32-bit	Position Command (Floats called out in VisualMotion program)*see Note				Multiplex, 32-bit	Axis Parameter 102 (Feedback Position)			
	Float 11	Float 21	Float 31	2		Parameter A-1.102	Parameter A-2.102	Parameter A-3.102	2
Multiplex, 32-bit	Counter (Integers called out in VisualMotion program)*see Note				Multiplex, 32-bit	Axis Parameter 112 (Feedback Velocity)			
	Integer 11	Integer 21	Integer 31	2		Parameter A-1.100	Parameter A-2.100	Parameter A-3.100	2
Multiplex, 32-bit	Dwell Time (Integers called out in VisualMotion program)*see Note				Multiplex, 32-bit	Maximum Attempts (Integers called out in VisualMotion program) *see Note			
	Integer 14	Integer 24	Integer 34	2		Integer 13	Integer 23	Integer 33	2
				TOTAL: 12 Data Words					TOTAL: 12 Data Words

Note: We suggest devising a system for assigning multiplex integers and floats to particular axes or data sets. (However, there is no limitation on how to divide multiplex objects, except size.)In our sample data, the system is as follows:
1st digit: Axis number
2nd digit: Purpose (e.g. position, counter, dwell)

For example: Integers in above sample data

3 4

1=Axis 1, 2=Axis 2, 3=Axis 3... 1=Counter, 4=Dwell Time

Table 2-4: Sample Cyclic Data (with Multiplexing)

Non-Cyclic Data

We will assume that the following data will be transferred non-cyclically for system initialization or setup. Remember, the direct-mapped non-cyclic data is intended for infrequent data transfer.

Note: See ***Table 1-3: Non-Cyclic (Direct-Mapped) DeviceNet*** Data Objects for data limitations.

Output Data (From Fieldbus Master to CLC Card)	Size (Data Words)	Size (Bits)	Input Data (From CLC Card to Fieldbus Master)	Size (Data Words)	Size (Bits)
Float 16	2	32	Float 5	2	32
Integer 7	2	32	Integer 8	2	32
Register 101	1	16			

Table 2-5: Sample Non-Cyclic Data (with Multiplexing)

Note: For other data types, such as diagnostic text messages, or S and P parameters, use the data exchange object.

STEP II: Configure Fieldbus Slave Card with Setup Information and Cyclic Data Object Lists Using the Fieldbus Mapper

Important: The drive must be set to Parameter Mode to configure and assign the data mapping for Fieldbuses (STEP II and STEP III).

1. In the VisualMotion "Data" Menu, select "Fieldbus Mapper."

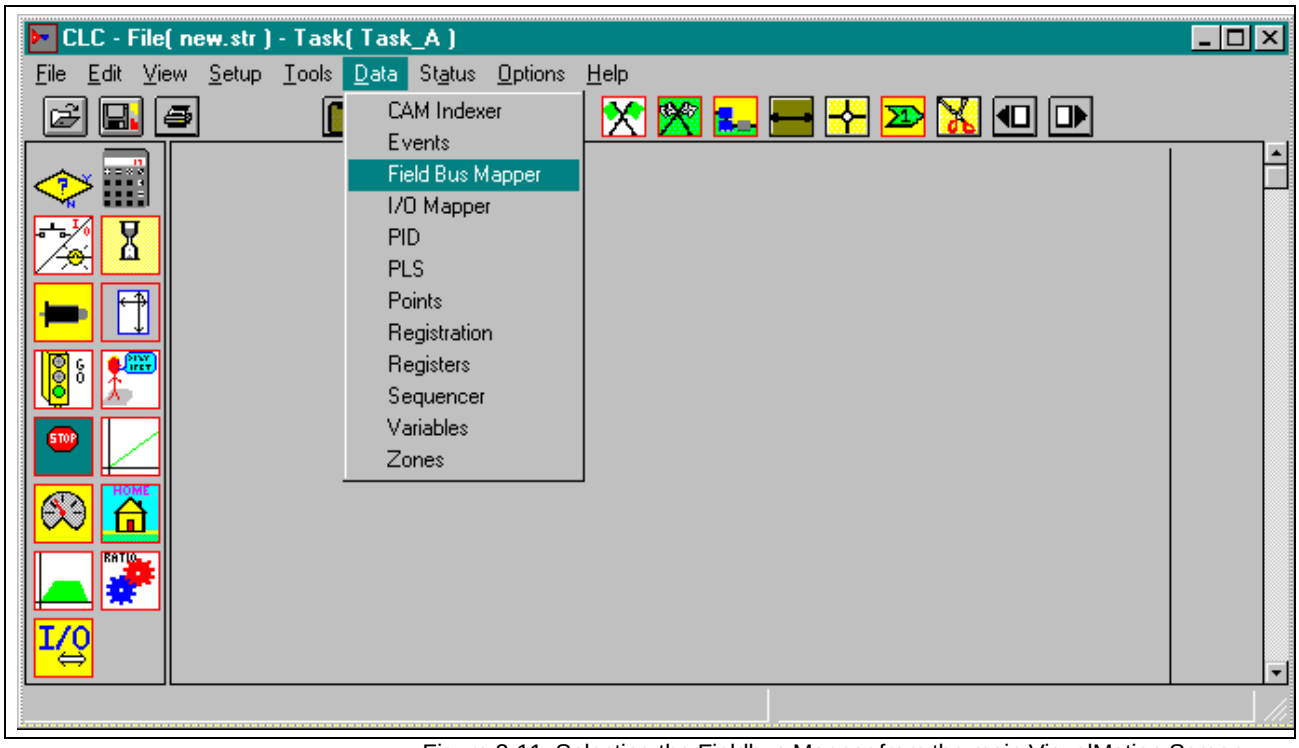


Figure 2-11: Selecting the Fieldbus Mapper from the main VisualMotion Screen

2. In the scroll box under "Bus Type," choose the desired bus (DeviceNet).
   The bus type may appear automatically if the Fieldbus card is connected and the firmware version is recognized.
3. Choose the cyclic data channel (Polled I/O).

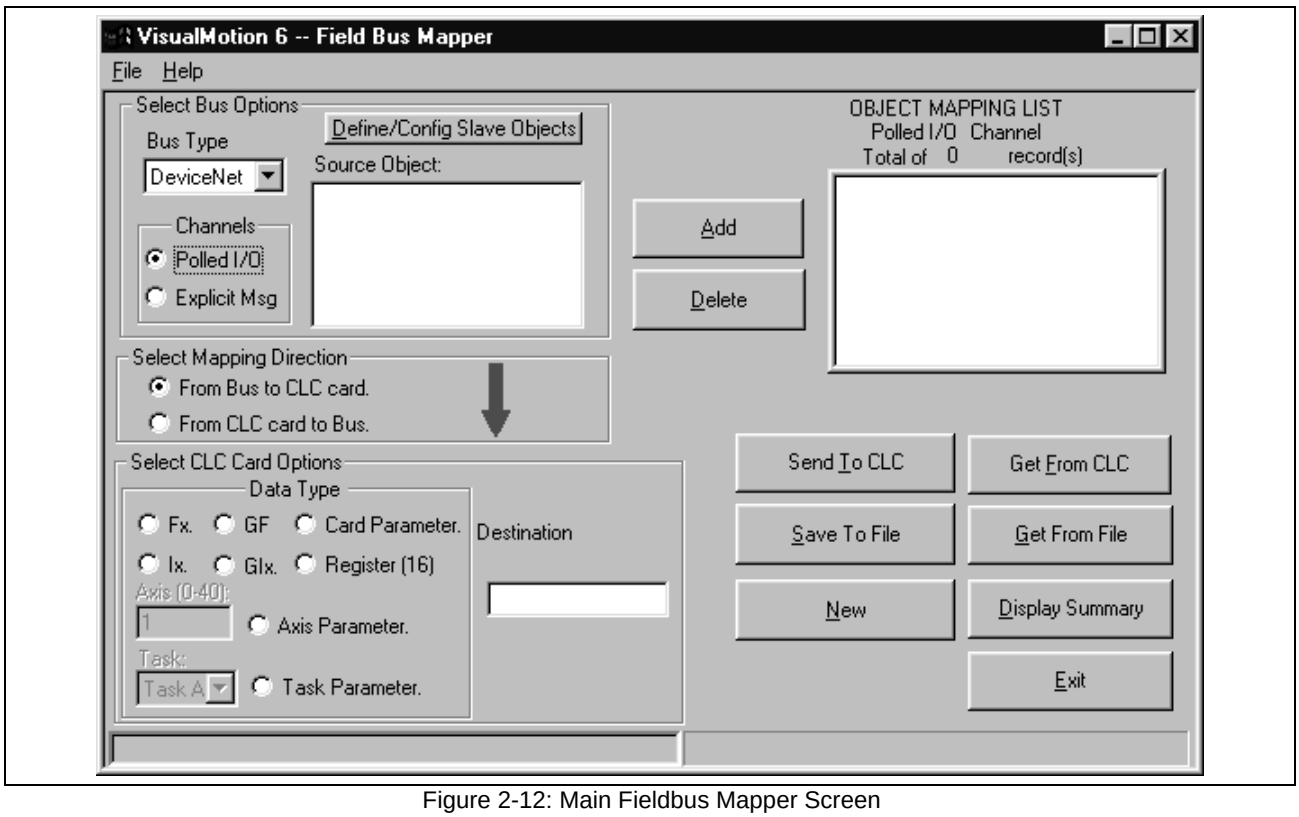


Figure 2-12: Main Fieldbus Mapper Screen

4. Click the button labeled "Define/Config Slave Objects." The Fieldbus Mapper will try to detect any configuration data from the CLC-D. If you are configuring a new list, click on the "NEW" button to clear any current data in the selected list. The DeviceNet Configuration Screen is pictured in ***Figure 2-13: DeviceNet Configuration Screen***.

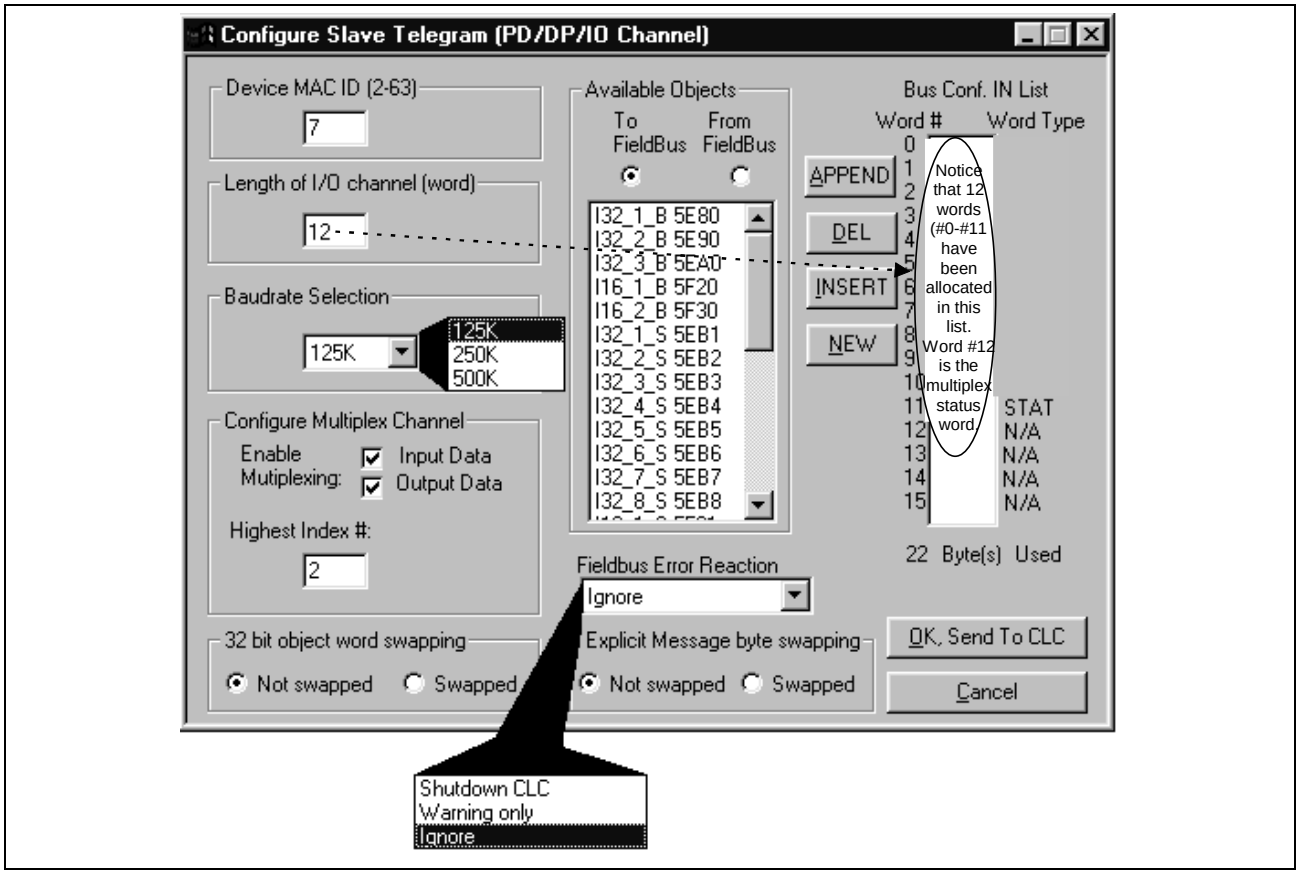


Figure 2-13: DeviceNet Configuration Screen

5. Set the device address to a unique number for the devices on the bus (2-99).
6. Set the "Length of I/O Channel (Word)" to the number of data words to be allocated for the input channel or the output channel, whichever is greater. According to the example data in ***Table 2-4: Sample Cyclic Data (with Multiplexing)*** under "STEP I: Determine the Cyclic and Non-Cyclic Data," we have set the length of the channel to 12 for the output and input channels.
7. Select the baud rate in the "Baudrate" combo box to match that of the other network devices.
8. Select "Not swapped" (default) or "Swapped" under "32 bit object word swapping," as required by your PLC. See ***Word and Byte Swapping*** on page 4-1.
9. Select "Not swapped" (default) or "Swapped" under "Explicit message byte swapping," as required by your PLC. See ***Word and Byte Swapping*** on page 4-1.
10. Click in both checkboxes next to "Enable Multiplexing," because we want to configure multiplex objects in both the output and input channels.

    Notice that the 13th word (word #12) shows the status/control word (STAT in the Bus Conf. IN list, CNTL in the Bus Conf. OUT list). The bus now actually has 13 words in each of the bus configuration lists: the 12 words that were set on the left, plus the status/control word. See ***Multiplex Control and Status Words*** on page 1-5 for information about the usage of this extra word when accessing data from the Fieldbus master (PLC).
11. Fill in the highest index number as "2," because we are using only 3 sets of multiplex data (Indexes 0-2, for axes 1-3). The index number will always be one less than the number of indices, because the first index number is "0."
12. Ensure that the combo box under "Fieldbus Error Reaction" is currently set to "Ignore." Any other setting during setup will cause repeated error messages. Set the desired Error Reaction when the machine is ready for commissioning. See **Fieldbus Error Reaction** on page 3-9 for detailed information about the possible settings.
13. Choose the radio button below the words "To FieldBus."
14. Click “NEW” to clear up any existing data in the current list.

Note: The desired order of the data on the master determines the object types (16- or 32-bit) and order of objects you place in the bus configuration list. Therefore, care should be taken when placing these objects.

---

15. For each data type desired, choose a corresponding available object in the list on the left and click "APPEND" to place it at the end of the "Bus Conf. IN List."
    Each available object has a typecode to identify its size and type.
    Figure 2-14: Configuring the DeviceNet Configuration IN List contains a description of the typecode.

Following are descriptions of the buttons to manipulate the bus configuration lists:

APPEND	inserts an available object after the last word placed in the current list.
DEL	removes the selected object from the current list.
INSERT	inserts an available object above the selected object in the current list.
NEW	clears up the current list (only in the direction selected under “Available Objects”).

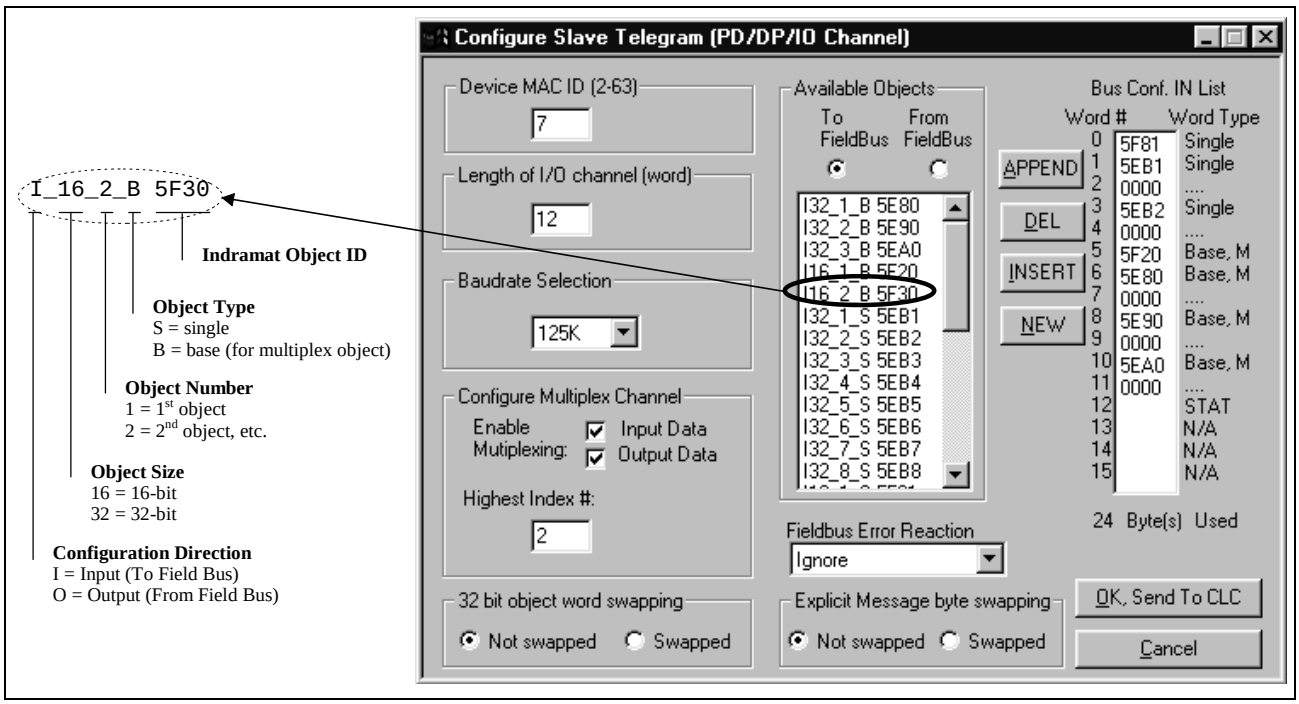


Figure 2-14: Configuring the DeviceNet Configuration IN List (with Multiplexing)

16. Choose the radio button below the words "From FieldBus."

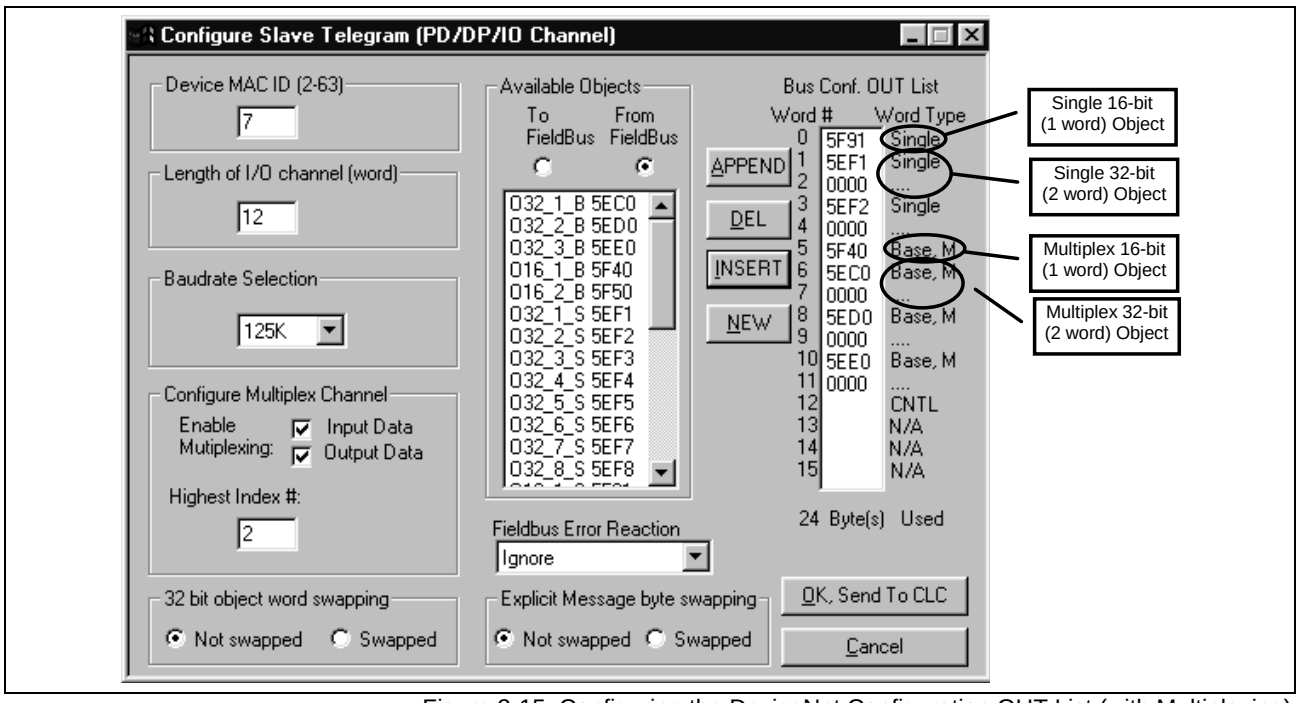


Figure 2-15: Configuring the DeviceNet Configuration OUT List (with Multiplexing)

17. For each data type desired, choose a corresponding available object in the list on the left and click "APPEND" to place it in the "Bus Conf. OUT List."

We have now allocated the data objects in the cyclic channel so the desired data can be transferred between the CLC-D card and the DeviceNet Fieldbus card.

Note: The non-cyclic data requires no configuration of objects in a list. These objects are addressed directly and preassigned as non-cyclic. They need only to be mapped to the CLC-D data.

---

18. Click "OK, Send to CLC." The following window appears:

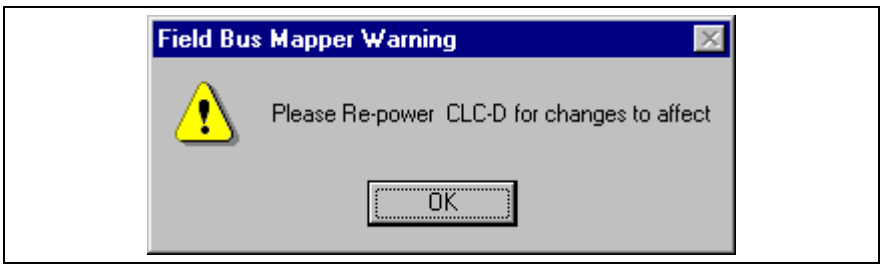


Figure 2-16: Warning after clicking "OK, Send to CLC"

STEP III: Define Cyclic Data Mapping Lists

STEP III associates the CLC-D data types to the objects assigned in STEP II. In our example, the objects to be added are:

Object Type	Output Data (From Bus to CLC Card)			Input Data (From CLC Card to Bus)		
Single	Register 100			Register 120		
"	Float 2			Global Float 22		
"	Integer 4			Global Integer 44		
Multiplex	Register 11	Register 12	Register 13	Register 31	Register 32	Register 33
"	Float 11	Float 21	Float 31	Parameter A-1.102	Parameter A-2.102	Parameter A-3.102
"	Integer 11	Integer 21	Integer 31	Parameter A-1.110	Parameter A-2.110	Parameter A-3.110
"	Integer 14	Integer 24	Integer 34	Integer 13	Integer 23	Integer 33

Table 2-6: Objects to be transferred cyclically (with multiplexing)

Adding Objects to the Cyclic Data Mapping List

1. Ensure that the designated bus type is correct.
2. Choose the desired cyclic data channel (Polled I/O).
3. Select the desired Mapping Direction.
   Our example begins with "From Bus to CLC Card."
4. Select the Data Type.
   In our example, the first single item to be added to this list is Register 120. Select "Register" as the data type.

Note: The CLC-D will show only the objects associated with the chosen data type in the Source Object list. For example, if you have selected the mapping direction "From Bus to CLC Card" and designate "Register" as the data type, only 16-bit output objects that have been mapped in the bus configuration list (see STEP II) will be shown in the Source Object list.

5. Select or fill in the Destination.
   In our example, we must select Register 120 from the list in the pop-up screen shown in ***Figure 2-7: Sample Selection List***, OR type 120 as the destination.

Note: The Fieldbus Mapper selection lists display only the labels of the currently active program.

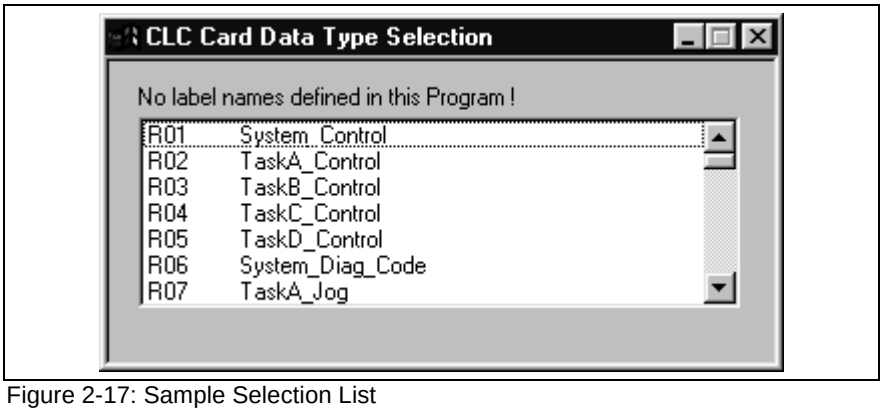


Figure 2-17: Sample Selection List

6. The source objects that are applicable to this data type will appear in the Source Object list.
   Choose the first available object.
7. Click the "Add" button.
   The object will appear in the "OBJECT MAPPING LIST" at the right.

Note: This step associates the object assigned in STEP II to the CLC-D data. The CLC-D scans the list created in this step and executes each mapping instruction **in order** every third SERCOS update cycle. Multiplex data can only be updated when this index is commanded by the multiplex control word.

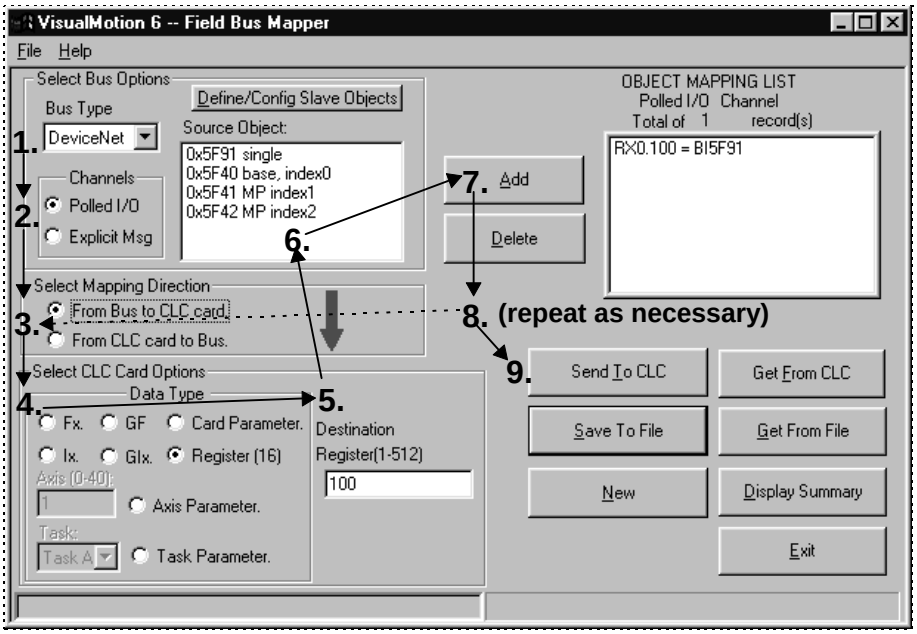


Figure 2-18: Adding Objects to the Cyclic Object Mapping List

Important: Cyclic data uses the objects configured in STEP II.

8. Repeat steps 3 through 7 above until each of the desired cyclic data objects has been added to the OBJECT MAPPING LIST.

Note: When mapping a PLC output object to a parameter, the parameter must be of the type "Read/Write Any Time." For parameters that are "Read Any Time/Write in Parameter Mode" or "Read Only," an error message is generated when you click “Send to CLC” if you try to map the parameters to a PLC output object. Examples: C-0-1550 (Read Any Time / Write in Parameter Mode) and C-0-1518 (Read Only).

Note: If you want to insert, replace or delete items in the Object Mapping List, double-click on the item in question. A new set of buttons appears to the left of the Object Mapping List box. See ***Changing an Existing Object Mapping List*** on page 2-8 for a detailed explanation of each button.

9. Click “Send to CLC” to save the selected object mapping list to the CLC-D card. Use any of the following buttons, as needed.

Saves the currently selected object mapping list to the CLC-D Card (to card parameter C-0-2600 if the cyclic [Polled I/O] channel is selected, and C-0-2700 if the non-cyclic [Explicit Messaging] channel is selected).

Saves the currently selected object mapping list to a file (with a .prm extension).

Clears up the current object mapping list.

Gets the currently selected object mapping list from the CLC-D Card (from card parameter C-0-2600 if the cyclic [Polled I/O] channel is selected, and C-0-2700 if the non-cyclic [Explicit Messaging] channel is selected).

Allows the user to open a previously saved file (with a .prm extension) of an object mapping list. When the file is opened, displays that object mapping list.

Generates an on-line summary report of the currently selected Fieldbus. This report can be printed by selecting "Print" from the File menu in the Fieldbus Mapper window.

Exits the Fieldbus Mapper utility.

Important: Ensure that your system is set to Parameter Mode for the changes to take place.

STEP IV: Define Non-Cyclic Data Mapping Lists (Direct Mapping)

The non-cyclic data uses objects that are pre-defined for this type of data.

In our example, the following data will be transmitted non-cyclically:

Output Data (From Bus to CLC Card)	**Input Data (From CLC Card to Bus)**
Float 16	Float 5
Integer 7	Integer 8
Register 101	

Table 2-7: Objects to be transferred non-cyclically

Adding Objects to the Non-Cyclic Data Mapping List (see *Figure 2-19: Adding Objects to the Non-Cyclic Object Mapping List*)

1. Ensure that the designated bus type is correct.
2. Choose the non-cyclic data channel (Explicit Messaging).
3. Select the desired Mapping Direction.
   In our example, we will begin with the output list, or "From Bus to CLC."
4. Select the Data Type.
   In our example, the first item to be added to this list is Float 16. Select "Float" as the data type.
5. Fill in the Destination.
   In our example, we must type 16 as the destination.
6. The source objects that are applicable to this data type will appear in the Source Object list. Choose the first available object.
7. Click the "Add" button.
   The object will appear in the "OBJECT MAPPING LIST" at the right. This list is scanned by the CLC only when a non-cyclic request comes in and executes the mapping.
8. Repeat steps 2 through 7 above until each of the desired non-cyclic objects has been added to the OBJECT MAPPING LIST in both directions ("From Bus to CLC card" AND "From CLC card to Bus").

Note: If you want to insert, replace or delete items in the Object Mapping List, double-click on the item in question. A new set of buttons appears to the left of the Object Mapping List box. See ***Changing an Existing Object Mapping List*** on page 2-8 for a detailed explanation of each button.

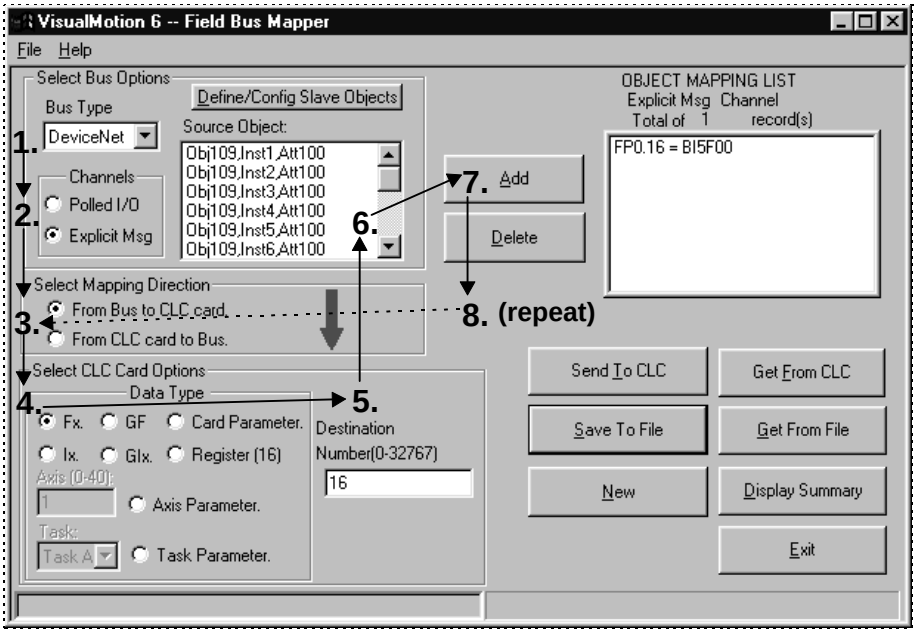


Figure 2-19: Adding Objects to the Non-Cyclic Object Mapping List

9. Click “Send to CLC” to save the selected object mapping list to the CLC-D card. Use any of the following buttons, as needed.

Saves the currently selected object mapping list to the CLC-D Card (to card parameter C-0-2600 if the cyclic [Polled I/O] channel is selected, and C-0-2700 if the non-cyclic [Explicit Messaging] channel is selected).

Saves the currently selected object mapping list to a file (with a .prm extension).

Clears up the current object mapping list.

Gets the currently selected object mapping list from the CLC-D Card (from card parameter C-0-2600 if the cyclic [Polled I/O] channel is selected, and C-0-2700 if the non-cyclic [Explicit Messaging] channel is selected).

Allows the user to open a previously saved file (with a .prm extension) of an object mapping list. When the file is opened, displays that object mapping list.

Generates an on-line summary report of the currently selected Fieldbus. This report can be printed by selecting "Print" from the File menu in the Fieldbus Mapper window.

Exits the Fieldbus Mapper utility.

Note: For debugging purposes, the Fieldbus Mapper follows each mapped item with the cursor as it is saved to the CLC-D. If there is a problem with the mapping of a particular object, an error message appears while the cursor remains on the object.

10. **Mapping is complete!**
For programming information, refer to ***Information for the GPS Programmer*** on page 3-1or ***Information for the PLC Programmer*** on page 4-1.

3 Information for the GPS Programmer

3.1 Viewing/Printing a Summary Report of the Current Fieldbus (only when CLC-D is on-line)

To View a Summary Report:

From the Fielbus Mapper Window, click on the

button or select "Display Summary" from the File menu.

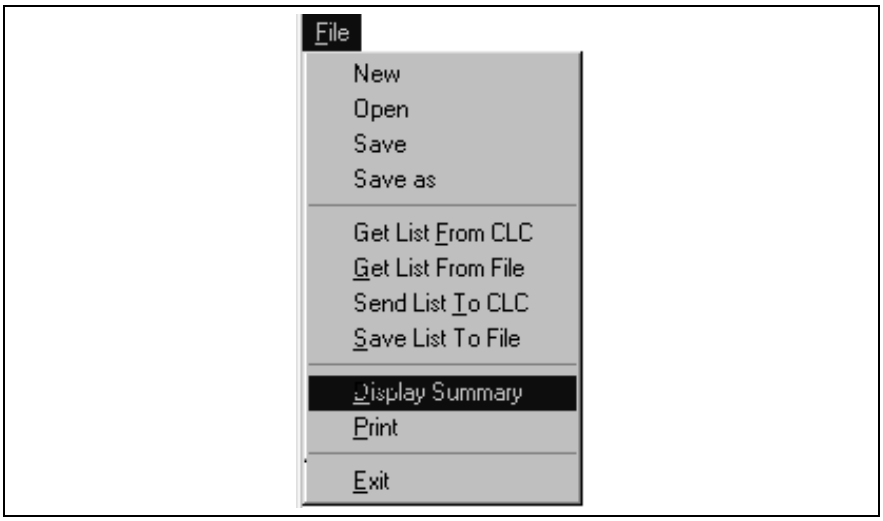


Figure 3-1: File Menu → Display Summary

A scrollable window will appear:

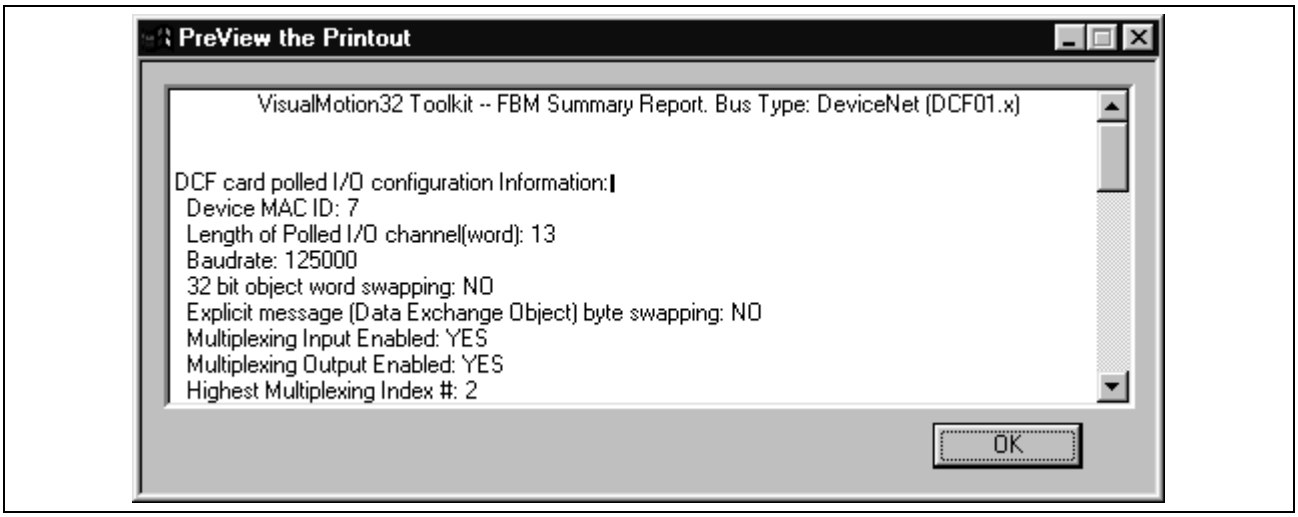

PreView the Printout
VisualMotion32 Toolkit -- FBM Summary Report. Bus Type: DeviceNet (DCF01.x)
DCF card polled I/O configuration Information:
Device MAC ID: 7
Length of Polled I/O channel(word): 13
Baudrate: 125000
32 bit object word swapping: NO
Explicit message (Data Exchange Object) byte swapping: NO
Multiplexing Input Enabled: YES
Multiplexing Output Enabled: YES
Highest Multiplexing Index #: 2
OK

Figure 3-2: Summary Report View Window

To Print a Summary Report:

From the Fieldbus Mapper window, select "Print" from the File menu.

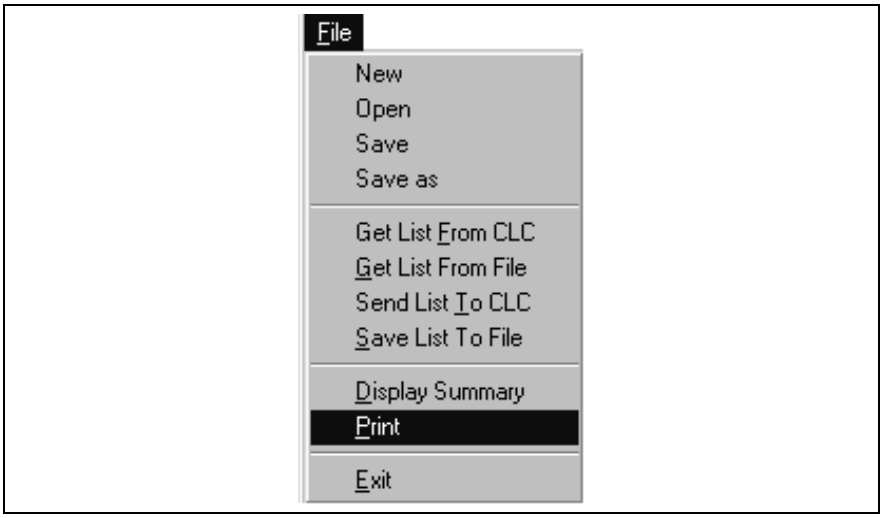


Figure 3-3: File Menu → Print

VisualMotion32 Toolkit – FBM Summary Report. Bus Type: Profibus (DPF 05.x)

Date: 05/05/99 Page 01 Time: 11:18:35

DCF card polled I/O configuration Information:

Device MAC ID: 7
Length of Polled I/O channel (word): 13
Baudrate: 125000
32 bit object word swapping: NO
Explicit message (Data Exchange Object) byte swapping: NO
Multiplexing Input Enabled: YES
Multiplexing Output Enabled: YES
Highest Multiplexing Index #: 2

Bus Confg. Input List (Polled I/O)

Word#:	Object#:	Object Type
00	0x5F81	16 bit, single
01	0x5EB1	32 bit, single
02	0x0000	.....
03	0x5EB2	32 bit, single
04	0x0000	.....
05	0x5F20	16 bit, baseMP
06	0x5E80	32 bit, baseMP
07	0x0000	.....
08	0x5E90	32 bit, baseMP
09	0x0000	.....
10	0x5EA0	32 bit, base MP
11	0x0000	.....
12	0x5FFD	MP StatusWord

Bus Confg. Output List (Polled I/O)

Word#:	Object#:	Object Type
00	0x5F91	16 bit, single
01	0x5EF1	32 bit, single
02	0x0000	.....
03	0x5EF2	32 bit, single
04	0x0000	.....
05	0x5F40	16 bit, baseMP
06	0x5EC0	32 bit, baseMP
07	0x0000	.....
08	0x5ED0	32 bit, baseMP
09	0x0000	.....
10	0x5EE0	32 bit, baseMP
11	0x0000	.....
12	0x5FFC	MP ControlWord

CLC/GPS data mapping Lists for DCF card Polled I/O:

Data mapping IN List (Polled I/O)

Word#:	Data Type	Object Type
00	RX0.100	16 bit, single
01, 02	HP0.22	32 bit, single
03,04	GP0.44	32 bit, single
05	RX0.31	16 bit, mult00
	RX0.32	16 bit mult01
	RX0.33	16 bit, mult02
06, 07	AP1.102	32 bit, mult00
	AP2.102	32 bit, mult01
	AP3.102	32 bit, mult02
08, 09	AP1.112	32 bit, mult00
	AP2.112	32 bit, mult01
	AP3.112	32 bit, mult02
10, 11	AP1.142	32 bit, mult00
	AP2.142	32 bit, mult01
	AP3.142	32 bit, mult02
12	0x5FFD	MP StatusWord

Data mapping OUT List (Polled I/O)

Word#:	Data Type	Object Type
00	RX0.120	16 bit, single
01, 02	FP0.2	32 bit, single
03, 04	IP0.4	32 bit, single
05	RX0.11	16 bit, mult00
	RX0.12	16 bit, mult01
	RX0.13	16 bit, mult02
06, 07	FP0.11	32 bit, mult00
	FP0.21	32 bit, mult01
	FP0.31	32 bit, mult02
08, 09	IP0.11	32 bit, mult00
	IP0.21	32 bit, mult01
	IP0.31	32 bit, mult02
10, 11	IP0.14	32 bit, mult00
	IP0.24	32 bit, mult01
	IP0.34	32 bit, mult02
12	0x5FFC	MP ControlWord

CLC/GPS data mapping Lists for DCF card Explicit Message (Direct Mapped Objects):

Data mapping IN List (EXP. Message)

Object#:	Data Type	Object Type
Obj110,Inst1,Att100	FP0.5	32 bit
Obj110,Inst2,Att100	IP0.8	32 bit

Data mapping OUT List (EXP. Message)

Object#:	Data Type	Object Type
Obj109,Inst1,Att100	FP0.16	32 bit
Obj109,Inst2,Att100	IP0.7	32 bit
Obj116,Inst1,Att100	RX0.121	16 bit

Figure 3-4: Printout of Sample Fieldbus Mapping

3.2 Fieldbus-Accessible Parameters

The following tables contain all of the Fieldbus-accessible parameters. Read and write access to these parameters is listed for both cyclic and non-cyclic data.

Parameter		Cyclic		Non-Cyclic (Direct-Mapped)	
Type/No.	Description	Read	Write	Read	Write
A-0-0020	Maximum Velocity	X (3)		X (3)	X (3)
A-0-0021	Maximum Acceleration	X		X	X
A-0-0022	Maximum Deceleration	X		X	X
A-0-0023	Jog Acceleration	X		X	X
A-0-0025	Maximum Jog Increment	X		X	X
A-0-0026	Maximum Jog Velocity	X		X	X
A-0-0031	CLC Cam/Ratio Master Factor (N)	X		X	
A-0-0032	CLC Cam/Ratio Slave Factor (M)	X		X	
A-0-0033	CLC Cam Stretch Factor (H)	X		X	
A-0-0034	CLC Cam Currently Active	X		X	X
A-0-0035	CLC Cam Position Constant (L)	X		X	
A-0-0037	Ratio Mode Step Rate	X		X	X
A-0-0038	Ratio Mode Options	X		X	
A-0-0100	Target Position	X (1, 3)		X (3)	
A-0-0101	Command Position	X (1, 3)		X (3)	
A-0-0102	Feedback Position	X (1, 3)		X (3)	
A-0-0110	Programmed Velocity	X (1, 3)	X (1, 3)	X (3)	X (3)
A-0-0111	Commanded Velocity	X (1, 3)		X (3)	
A-0-0112	Feedback Velocity	X (1, 3)		X (3)	
A-0-0120	Programmed Acceleration	X (1, 3)		X (3)	
A-0-0141	Torque Command	X (1, 3)		X (3)	
A-0-0142	Torque Feedback	X (1, 3)		X (3)	
A-0-0150	Programmed Ratio Adjust	X (1, 3)		X (3)	
A-0-0151	Programmed Phase Offset	X (3)		X (3)	
A-0-0153	CLC Phase Adjust Average Velocity	X (3)		X (3)	X (3)
A-0-0155	CLC Phase Adjust Time Constant	X		X	X
A-0-0157	Current Phase/CLC Cam Master Offset	X (3)		X (3)	
A-0-0159	Ratio Adjust Step Rate	X (3)		X (3)	X (3)
A-0-0160	Commanded Ratio Adjust	X (1, 3)		X (3)	
A-0-0164	ELS Options	X		X	
A-0-0180	Optional Command ID #1	X		X	
A-0-0181	Optional Command ID #2	X		X	
A-0-0182	Optional Command ID #3	X		X	
A-0-0185	Optional Feedback ID #1	X		X	
A-0-0186	Optional Feedback ID #2	X		X	
A-0-0190	Command Data #1	X (1, 2, 3)	X (1, 2, 3)	X (1, 2, 3)	X (1, 2, 3)
A-0-0191	Command Data #2	X (1, 2, 3)	X (1, 2, 3)	X (1, 2, 3)	X (1, 2, 3)
A-0-0192	Command Data #3	X (1, 2, 3)	X (1, 2, 3)	X (1, 2, 3)	X (1, 2, 3)
A-0-0195	Feedback Data #1	X (1, 2, 3)		X (1, 2, 3)	
A-0-0196	Feedback Data #2	X (1, 2, 3)		X (1, 2, 3)	
C-0-0001	Language Selection	X		X	
C-0-0002	Unit Number	X		X	X
C-0-0009	Error Reaction Mode	X		X	
C-0-0010	System Options	X		X	

1 = Must be configured by GPS in SERCOS telegram to update data. Otherwise, unexpected GPS system errors may occur.

2 = The data in this parameter is ONLY interpreted as type Float.

3 = This parameter is valid for fieldbus mapping in firmware version 06v64 and later. The data is read/written via the fieldbus in SERCOS phase 4 only. This parameter requires additional GPS resources to transfer data to/from the fieldbus. Therefore, care must be taken when using it for fieldbus communications with consideration to GPS system resources. Monitor GPS Register 26 for fieldbus mapping resource consumption, and increase the SERCOS update (C-0-0099) if necessary.

4 = No limit-checking is performed when writing to this parameter via a fieldbus. Therefore, care must be taken to ensure the proper data is sent.

Table 3-1: Fieldbus-Accessible Parameters (part 1 of 3)

Parameter		Cyclic		Non-Cyclic (Direct-Mapped)	
Type/No.	Description	Read	Write	Read	Write
C-0-0016	Communication Time-Out Period	X		X	X
C-0-0020	Transmitter Fiber Optic Length	X		X	X
C-0-0021	User Watchdog Timer	X		X	
C-0-0022	User Watchdog Task ID	X	X (4)	X	X
C-0-0030	Option Cards Active	X		X	
C-0-0031	Option Card Status	X		X	
C-0-0042	World Large Increment	X		X	X
C-0-0043	World Small Increment	X		X	X
C-0-0045	World Fast Jog Speed	X		X	X
C-0-0046	World Slow Jog Speed	X		X	X
C-0-0052	Axis Large Increment	X		X	X
C-0-0053	Axis Small Increment	X		X	X
C-0-0055	Axis Fast Jog Velocity	X		X	X
C-0-0056	Axis Slow Jog Velocity	X		X	X
C-0-0090	Download Block Size	X		X	X
C-0-0120	Operating Mode	X		X	
C-0-0121	SERCOS Communication Phase	X		X	
C-0-0123	Diagnostic Code	X		X	
C-0-0125	System Timer Value	X (3)		X	X
C-0-0151	Master 1 Drive Address	X		X	
C-0-0153	Virtual Master Programmed Velocity	X (3)		X (3)	X (3)
C-0-0154	Virtual Master Programmed Acceleration	X (3)		X (3)	X (3)
C-0-0155	Virtual Master Programmed Deceleration	X (3)		X (3)	X (3)
C-0-0156	Virtual Master E-Stop Deceleration	X (3)		X (3)	X (3)
C-0-0157	ELS Master Current Position	X (3)		X (3)	X (3)
C-0-0158	ELS Master Current Velocity	X (3)		X (3)	
C-0-0159	Master 1 Zero Velocity Window	X (3)		X (3)	X (3)
C-0-0160	Virtual Master Max. Jog Velocity	X		X	X
C-0-0161	Master 1 Ratio Input	X		X	
C-0-0162	Master 1 Ratio Output	X		X	
C-0-0164	Master 1 Encoder Type	X		X	
C-0-0170	PLS 1 Mask Register	X		X	X
C-0-0801	Pendant Protection Level 1 Password	X		X	X
C-0-0802	Pendant Protection Level 2 Password	X		X	X
C-0-0803	Pendant User Accessible Floats Section	X		X	X
C-0-0804	Pendant User Accessible Integers Section	X		X	X
C-0-0805	Pendant Start of User Accessible Registers	X		X	X
C-0-0806	Pendant End of User Accessible Registers	X		X	X
C-0-0807	Pendant Password Timeout	X		X	X
C-0-0810	TPT Message and Prompt Control Word	X	X (4)	X	X
C-0-0811	User Task Controlled Menu ID for TPT	X	X (4)	X	X
C-0-0812	User Task Controlled Task ID for TPT	X	X (4)	X	X
C-0-0813	User Task Controlled Axis Number for TPT	X	X (4)	X	X
C-0-0814	TPT Data Transaction Word	X	X (4)	X	X
C-0-1001	ELS Secondary Master	X		X	
C-0-1010	Master Switching Threshold	X (3)		X (3)	X (3)

1 = Must be configured by GPS in SERCOS telegram to update data. Otherwise, unexpected GPS system errors may occur.

2 = The data in this parameter is ONLY interpreted as type Float.

3 = This parameter is valid for fieldbus mapping in firmware version 06v64 and later. The data is read/written via the fieldbus in SERCOS phase 4 only. This parameter requires additional GPS resources to transfer data to/from the fieldbus. Therefore, care must be taken when using it for fieldbus communications with consideration to GPS system resources. Monitor GPS Register 26 for fieldbus mapping resource consumption, and increase the SERCOS update (C-0-0099) if necessary.

4 = No limit-checking is performed when writing to this parameter via a fieldbus. Therefore, care must be taken to ensure the proper data is sent.

Table 3-2: Fieldbus-Accessible Parameters (part 2 of 3)

Parameter		Cyclic		Non-Cyclic (Direct-Mapped)	
Type/No.	Description	Read	Write	Read	Write
C-0-1011	Diff. Between Real Master Positions	X (3)		X (3)	
C-0-1012	Master Synchronization Acceleration	X		X	
C-0-1013	Master Synchronization Time Constant	X		X	
C-0-1014	Master Synchronization Velocity Window	X		X	
C-0-1015	Virtual Master Current Position	X (3)		X (3)	X (3)
C-0-1016	Virtual Master Current Velocity	X (3)		X (3)	
C-0-1017	Position Difference Selection	X		X	X
C-0-1018	Position Monitoring Window	X (3)		X (3)	X (3)
C-0-1019	Position Difference	X (3)		X (3)	
C-0-1508	Master 1 Filter Cutoff Frequency	X		X	
C-0-1509	Master 1 Filter Type	X		X	
C-0-1517	Master 1 Current Position	X (3)		X (3)	X (3)
C-0-1518	Master 1 Current Velocity	X (3)		X (3)	
C-0-1550	Master 2 Type	X		X	
C-0-1551	Master 2 Drive Address	X		X	
C-0-1552	Master 2 Encoder Type	X		X	
C-0-1554	Master 2 Ratio Input	X		X	
C-0-1555	Master 2 Ratio Output	X		X	
C-0-1556	Master 2 Zero Velocity Window	X (3)		X (3)	X (3)
C-0-1558	Master 2 Filter Cutoff Frequency	X		X	
C-0-1559	Master 2 Filter Type	X		X	
C-0-1567	Master 2 Current Position	X (3)		X (3)	X (3)
C-0-1568	Master 2 Current Velocity	X (3)		X (3)	
C-0-2014	Start of SERCOS I-O Station Registers	X		X	
C-0-2015	Registers Allocated per I-O Station	X		X	
T-0-0002	Task Options	X		X	
T-0-0005	World Position Units	X		X	
T-0-0010	Kinematic Number	X		X	
T-0-0011	Coordinated X-Axis	X		X	
T-0-0012	Coordinated Y-Axis	X		X	
T-0-0013	Coordinated Z-Axis	X		X	
T-0-0020	Maximum Path Speed	X (3)		X (3)	
T-0-0021	Maximum Acceleration	X		X	
T-0-0022	Maximum Deceleration	X		X	
T-0-0023	Look Ahead Distance	X		X	X
T-0-0024	Velocity Override	X		X	X
T-0-0025	Maximum Jog Increment	X		X	X
T-0-0026	Maximum Jog Velocity	X		X	X
T-0-0035	Relative Point Used for Origin	X		X	X
T-0-0036	Relative Point Used for Tool Frame	X		X	X
T-0-0100	Target Point Number	X (3)		X (3)	
T-0-0101	Segment Status	X (3)		X (3)	
T-0-0102	Rate Limit Status	X (3)		X (3)	

1 = Must be configured by GPS in SERCOS telegram to update data. Otherwise, unexpected GPS system errors may occur.

2 = The data in this parameter is ONLY interpreted as type Float.

3 = This parameter is valid for fieldbus mapping in firmware version 06v64 and later. The data is read/written via the fieldbus in SERCOS phase 4 only. This parameter requires additional GPS resources to transfer data to/from the fieldbus. Therefore, care must be taken when using it for fieldbus communications with consideration to GPS system resources. Monitor GPS Register 26 for fieldbus mapping resource consumption, and increase the SERCOS update (C-0-0099) if necessary.

4 = No limit-checking is performed when writing to this parameter via a fieldbus. Therefore, care must be taken to ensure the proper data is sent.

Table 3-3: Fieldbus-Accessible Parameters (part 3 of 3)

3.3 Register 19 Definition (Fieldbus Status)

Diagnostic Object 5FF2

The diagnostic object 5ff2 holds the information for "Fieldbus Status," and this information is stored in VisualMotion Register 19. The register information can be referenced in a VisualMotion application program to respond to the status of each bit. The use of these bits is application-dependent.

Table 3-4: Bit Assignment for the Diagnostic Object 5ff2 (VisualMotion Register 19) contains the bit assignment for the diagnostic object 5ff2. The assigned bits are labeled with "x" and the bit number in the second row. Unassigned bits are labeled with "---."

16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
---	x15	---	x13	---	---	---	---	---	---	x6	x5	x4	---	x2	x1

Table 3-4: Bit Assignment for the Diagnostic Object 5ff2 (VisualMotion Register 19)

Bit Definitions

x1, x2 Status bits for the internal DPR (Dual-Port RAM) communication between the Fieldbus slave and the CLC-D:

x1: FB Init OK , LSB (least significant bit)

x2: FB Init OK, MSB (most significant bit)

The bit combinations for x1 and x2 are as follows:

Bit 2 (CLC-D)	Bit 1 (Fieldbus)	Description
0	0	A reset has been executed on the DPR, **or** neither the CLC-D card nor the Fieldbus card have initialized the DPR.
0	1	The DPR is initialized by the Fieldbus card, but **not** yet by the CLC-D card.
1	0	The DPR initialization is complete. DPR has been initialized by the Fieldbus card and CLC-D card.
1	1	Fieldbus to CLC-D communications system is ready. The system looks for this combination of bit settings to initiate any communication between the Fieldbus and CLC-D cards. The watchdog-function via DPR communications with the Fieldbus slave works correctly.

Table 3-5: Possible Settings for Bits 1 and 2, Status Bits for DPR Communication

x4 Status bit for the active bus capabilities of the Fieldbus slaves (FB Slave Ready)

This bit is monitored for the Fieldbus Error Reaction. Whenever this bit goes to 0 after a Fieldbus card was initially found by the CLC-D, the selected Error Reaction (system shutdown, error message, or ignore) is initiated. See **Fieldbus Error Reaction** on page 3-9 for detailed information about the possible settings.

0--> The Fieldbus slave is not (yet) ready for data exchange.

1--> The Fieldbus slave can actively participate on the bus.

x5 Status bit for the non-cyclic channel (Explicit Messaging) (Non-Cyc Ready)

0--> The non-cyclic (Explicit Messaging) channel can not (yet) be used.

1--> The non-cyclic (Explicit Messaging) channel is ready for use by the Fieldbus Master.

x6 Status-bit for the reconfiguration of the cyclic (Polled I/O) channel (nCyc Chan Ready):

0--> The cyclic (Polled I/O) channel on the Fieldbus-card is configured properly. The system looks for this bit to be 0 before allowing data transfer.

1--> The cyclic (Polled I/O) channel is being reconfigured on the Fieldbus Card at this time.

x13 Status bit for non-supported GPS firmware (nVM FW OK):

0--> The GPS firmware version is supported with this Fieldbus interface.

1--> The GPS-firmware version is NOT supported by this Fieldbus interface card. The CLC-D hardware could support the Fieldbus interface, but the firmware version is not recognized as valid by the Fieldbus card firmware.

x15 Status bit for the cyclic data output (Cyclic Data Valid):

0--> The cyclic data outputs (coming in to the CLC-D) are INVALID.

1--> The cyclic data outputs (coming in to the CLC-D) are VALID. The system looks for this bit to be 1 before allowing data transfer.

3.4 Register 20 Definition (Fieldbus Diagnostics)

Diagnostic Object 5FF0

The diagnostic object 5ff0 holds the information for "Fieldbus Diagnostics," and this information is stored in VisualMotion Register 20.

Table 3-6: Bit Assignment for the Diagnostic Object 5ff0 (VisualMotion Register 20) contains the bit assignment for the diagnostic object 5ff0. The assigned bits are labeled with "x" and the bit number in the second row. Unassigned bits are labeled with "---."

16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
---	x15	x14	x13	---	---	---	---	x8	x7	x6	x5	x4	x3	x2	x1

Table 3-6: Bit Assignment for the Diagnostic Object 5ff0 (VisualMotion Register 20)

Bit Definitions

x1 - x8 Fieldbus Interface-specific fault code (FB Card Fault Code)

Note: This function is currently not available.

x13 - x15 Identification of the Fieldbus interface card (FB Card Found)

The bit combinations for x13, x14 and x15 are as follows:

Bit 15	Bit 14	Bit 13	Fieldbus Type
0	0	0	<NO CARD>
0	0	1	<Not Defined>
0	1	0	Interbus (DBS03)
0	1	1	DeviceNet (DCF01)
1	0	0	Profibus (DPF05)
1	0	1	CanOpen (DCF01)
1	1	0	<Not Defined>
1	1	1	<Not Defined>

Table 3-7: Identification of the Fieldbus Interface

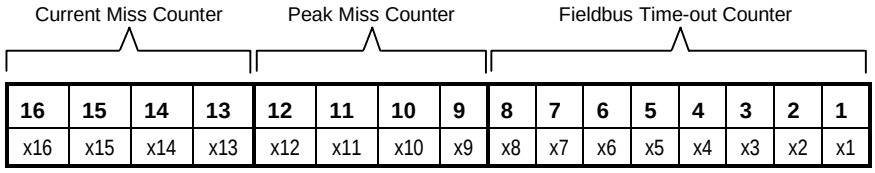
3.5 Register 26 Definition (Fieldbus Resource Monitor)

The "Fieldbus Resource Monitor" in register 26 can be used as a method for monitoring the attempts made to process the Object Mapping List parameter C-0-2600 across the Dual-port RAM. If after three SERCOS update cycles, the Object Mapping List is not successfully transmitted, a "**miss**" is noted.

Register 26 is divided into the following three counter types:

- Current Miss Counter.
- Peak Miss Counter.
- Fieldbus Timeout Counter.

Table 3-8: Bit Assignment for Fieldbus Resource Monitor (Register 26) contains the bit assignment for the Fieldbus counters. The assigned bits are labeled with an "x" followed by the bit number.



16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
x16	x15	x14	x13	x12	x11	x10	x9	x8	x7	x6	x5	x4	x3	x2	x1

Table 3-8: Bit Assignment for Fieldbus Resource Monitor (Register 26)

Note: View register 26 in Hexadecimal format to more easily monitor the Fieldbus counters.

Hex display format of register 26

0x0000

Fieldbus Time-out Counter (00-FF)
Increments by one every time 10 cyclic mapping list attempts are missed in 10 attempts.

Peak Miss Counter (0-9)
Displays the peak value of the Current Miss Counter.

Current Miss Counter (0-9)
Displays the count of missed cyclic mapping list attempts out of the last 10 attempts

Note: To view registers in hex, select ***Data ⇒ Registers*** within VisualMotion Toolkit. If the registers are not currently viewed in hex format, select ***Format ⇒ Hex*** in the *Active Program, Register* window.

Bit Definitions

x13 - x16 Status bits for the "Current Miss Counter."

An attempt is made to transmit the cyclic Object Mapping List, C-0-2600, across the Dual-port RAM every third SERCOS update cycle. For every 10 mapping list update attempts (30 SERCOS update cycles), the failed attempts are counted and displayed in these bits (values can range from 0-9). If 10 out of 10 mapping list updates attempts are missed, the "Fieldbus Timeout Counter" is incremented by one. This is an indication of a Fieldbus Mapping Timeout Error.

x9 - x12 Status bits for the "Peak Miss Counter."

These bits monitor the "Current Miss Counter's" peak count between a value from 0-9 and holds that value until a larger count is encountered.

x1 - x8 Status bits for the "Fieldbus Timeout Counter."

The count of these bits increments by one every time the "Current Miss Counter" encounters 10 out 10 missed attempts of the cyclic mapping list update. A count incremented by one represents a Fieldbus Mapping Timeout Error and is process by GPS according to the selected "Fieldbus Error Reaction" in parameter C-0-2635. See Error! Reference source not found. for an explanation of the Fieldbus Error Reaction setting.

Note: The GPS programmer can monitor the "Current Miss Counter" and define a custom error reaction for missed mapping list update attempts less than 10.

Note: The values in register 26 are read/write and can be reset by the user.

3.6 Fieldbus Error Reaction

Note: The Fieldbus Error Reaction setting is active only in SERCOS Phase 4. In all other SERCOS phases, it will be inactive.

You can choose how you would like the CLC-D system to react in case of a Fieldbus error. This reaction can be set in the "Configure Slave Telegram" screen, using the combo box labeled "Fieldbus Error Reaction."

Three options are available for the Error Reaction setting. Depending on the selected setting, the value 0, 1, or 2 is stored in Parameter C-0-2635:

Setting	Value in Parameter C-0-2635
Shutdown CLC	0 (default)
Warning Only	1
Ignore	2

Table 3-9: Parameter C-0-2635 Values for Error Reaction Settings

Fieldbus Mapper Timeout The Fieldbus Mapper continually scans the system for sufficient resources to process the cyclic data mapping list (2600-list). If 10 out of 10 attempts of the mapping list update are missed, the system is

considered to have insufficient resources and the selected error reaction is evoked, as follows:

If "Shutdown CLC" (0) is set in Parameter C-0-2635, the following error is generated from the CLC-D card: **520 Fieldbus Mapper Timeout**

If "Warning Only" (1) is set in Parameter C-0-2635, the following error is generated: **209 Fieldbus Mapper Timeout**

If "Ignore" (2) is set in Parameter C-0-2635, the system will update as resources become available, but there is no way to monitor whether or not updates actually occur.

Lost Fieldbus Connection

Object 5ff2, the dual-port RAM location on the Fieldbus card, indicates the status of the Fieldbus. Register 19 Bit 4, the Fieldbus Slave Ready Bit on the CLC card, mirrors object 5ff2. See ***3.3 Register 19 Definition (Fieldbus Status)*** for more specific bit information. The system monitors this bit and evokes the selected error reaction if the bit is low (0), after a Fieldbus card is found. A typical situation that will cause this condition is the disconnection of the Fieldbus cable from the DCF card.

If "Shutdown CLC" (0) is set in Parameter C-0-2635, the following error is generated from the CLC-D card (active in SERCOS Phase 4 only): **519 Lost Fieldbus Connection**

If "Warning Only" (1) is set in Parameter C-0-2635, the following error is generated (active in SERCOS Phase 4 only): **208 Lost Fieldbus Connection**

If "Ignore" (2) is set in Parameter C-0-2635, there is no noticeable reaction when Register 19 Bit 4 goes low, unless the GPS coding is customized to evoke a special reaction.

Troubleshooting Tip:

If a Fieldbus card is not found on the system, the Error Reaction setting will be ignored. If you have a Fieldbus card and the Error Reaction is not responding as expected, the system may not "see" your Fieldbus card.

4 Information for the PLC Programmer

4.1 *.eds File

Indramat supplies an *.eds file containing supporting information for the CLC-D with DCF01.x card DeviceNet slave configuration. Contact an Indramat technical representative for location of this file.

4.2 Word and Byte Swapping

In the Fieldbus Mapper, it is possible to enable automatic word and byte swapping on the DCF board (for both input and output), depending on the type of PLC used.

- **32-bit Object Word Swapping** - The setting of this option determines the order in which the two data words in **any 32-bit (double word) cyclic or non-cyclic direct-mapped object** are transmitted. The default setting, "Not swapped" causes the words to be transmitted in their usual order: [Word 1], [Word 2]. The "Swapped" setting causes the words to be transmitted in inverted order: [Word 2], [Word 1]. The setting of this option is stored in Card Parameter C-0-2636, bit 0.
- **Explicit Message Byte Swapping** - The setting of this option determines the order in which the bytes of **the non-cyclic data exchange object** are transmitted. The default setting, "Not swapped" causes the bytes to be transmitted in their usual order: [Byte 1], [Byte 2], [Byte 3], [Byte 4], ... The "Swapped" setting causes each pair of bytes to be transmitted in inverted order: [Byte 2], [Byte 1], [Byte 4], [Byte 3], ... The setting of this option is stored in Card Parameter C-0-2636, bit 1.

Example: Allen-Bradley SDN Module for SLC-Series PLC

When the Allen-Bradley SDN (DeviceNet Scanner) Module for the SLC-Series PLC is used, both **32-bit Object Word Swapping** and **Explicit Message Byte Swapping** can be set to "Swapped" in the Fieldbus Mapper, so the order of resulting data appears correctly.

4.3 Multiplex Data Bits in the Control and Status Words

Figure 4-1: Bit Definitions in Multiplex Control and Status Words contains diagrams of the bits contained in the control and status words and their bit definitions. The bits marked with ✖ are currently not used. The control word is appended as the last word of the cyclic data content, and the status word is appended as the slave response in the cyclic data input.

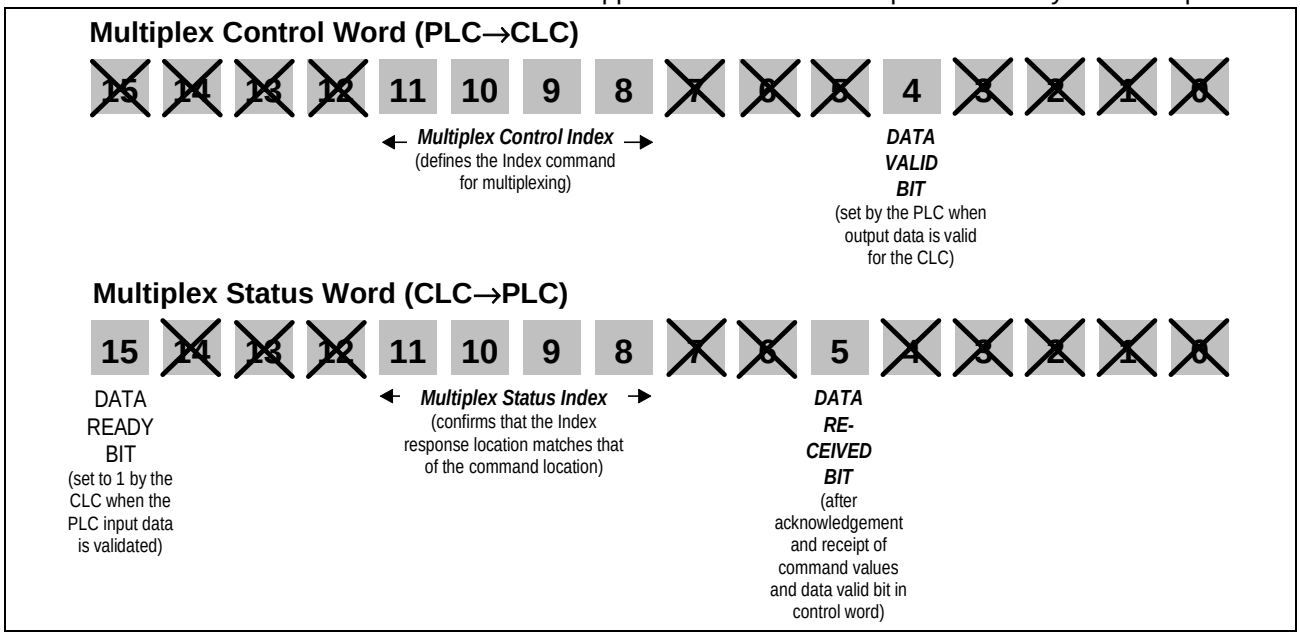

Multiplex Control Word (PLC→CLC)
15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
Multiplex Control Index
(defines the Index command for multiplexing)
DATA VALID BIT
(set by the PLC when output data is valid for the CLC)
Multiplex Status Word (CLC→PLC)
15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
DATA READY BIT
(set to 1 by the CLC when the PLC input data is validated)
Multiplex Status Index
(confirms that the Index response location matches that of the command location)
DATA RE-CEIVED BIT
(after acknowledgement and receipt of command values and data valid bit in control word)

Figure 4-1: Bit Definitions in Multiplex Control and Status Words

Figure 4-2: Write / Write and Read Process and ***Figure 4-3: Read-Only Process*** explain the exchange of multiplex information between the master (PLC) and the slave (CLC-D) using the control and status words:

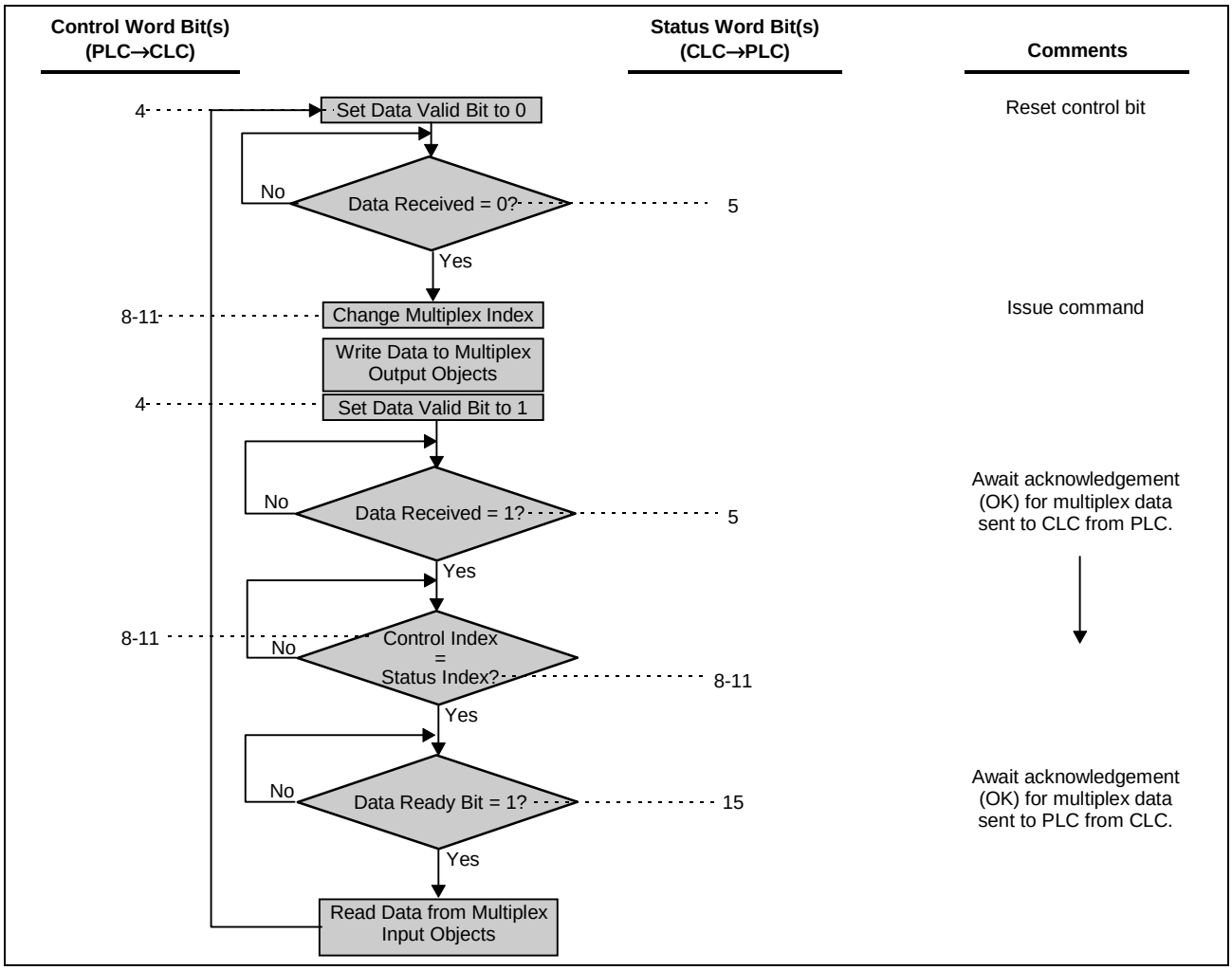


Figure 4-2: Write / Write and Read Process

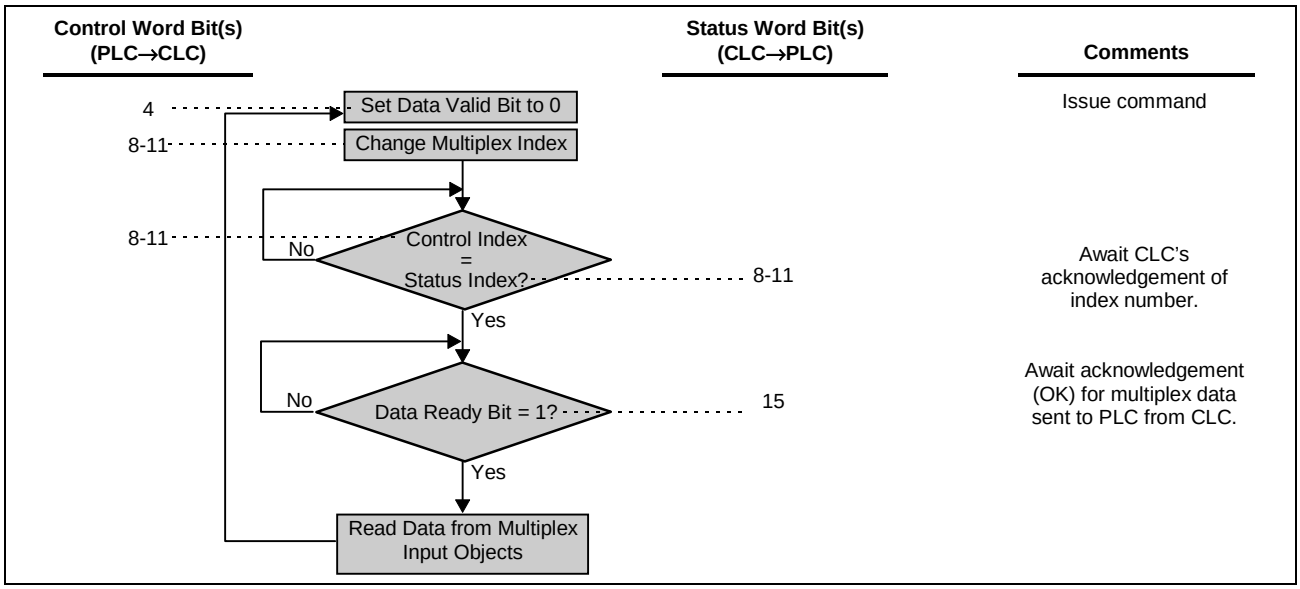


Figure 4-3: Read-Only Process

4.4 Non-Cyclic Transmission (Direct-Mapped Objects)

A number of objects (see ***Table 4-10: Non-Cyclic Data Objects*** below) can represent VisualMotion data types if they are mapped in the non-cyclic object mapping list (C-0-2700). This method provides a simple but inflexible method of transferring non-cyclic data from the master to the slave.

Number of Objects Available	Data Size	Direction	Numeric Names of Objects
16	32-bit	in	Class 110 Instance 1-16 Attribute 100
16	32-bit	out	Class 109 Instance 1-16 Attribute 100
32	16-bit	in	Class 115, 119 Instance 1-16 Attribute 100
32	16-bit	out	Class 116, 120 Instance 1-16 Attribute 100

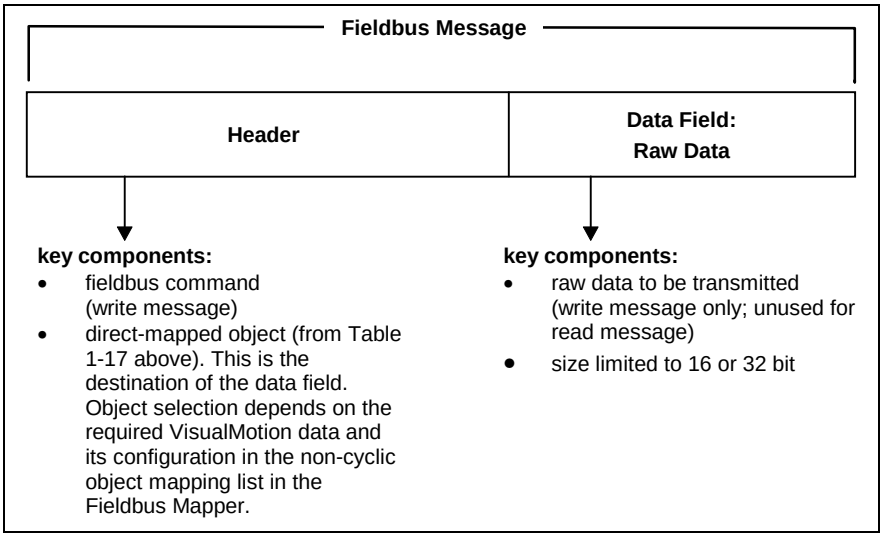
Table 4-10: Non-Cyclic Data Objects

Selecting a Direct-Mapped Object

The VisualMotion data available for non-cyclic direct-mapped objects can be viewed and printed via the summary report function in the VisualMotion Fieldbus Mapper tool (see ***Viewing/Printing a Summary Report of the Current Fieldbus*** on page 3-1).

Transmission Sequence via a Direct-Mapped Object

Note: For the direct-mapped object, only one transmission (and one response) is required, to send a read or write message to the CLC-D card and to receive a response from the CLC-D card.


Fieldbus Message
Header
Data Field:
Raw Data
key components:
• fieldbus command (write message)
• direct-mapped object (from Table 1-17 above). This is the destination of the data field. Object selection depends on the required VisualMotion data and its configuration in the non-cyclic object mapping list in the Fieldbus Mapper.
key components:
• raw data to be transmitted (write message only; unused for read message)
• size limited to 16 or 32 bit

Important: The format of the Fieldbus message header and the method of implementation are dependent on the Fieldbus type and the master (PLC) being used. Refer to your Fieldbus master/PLC documentation for proper transport and formatting of the message header.

Non-Cyclic Direct-Mapped Write

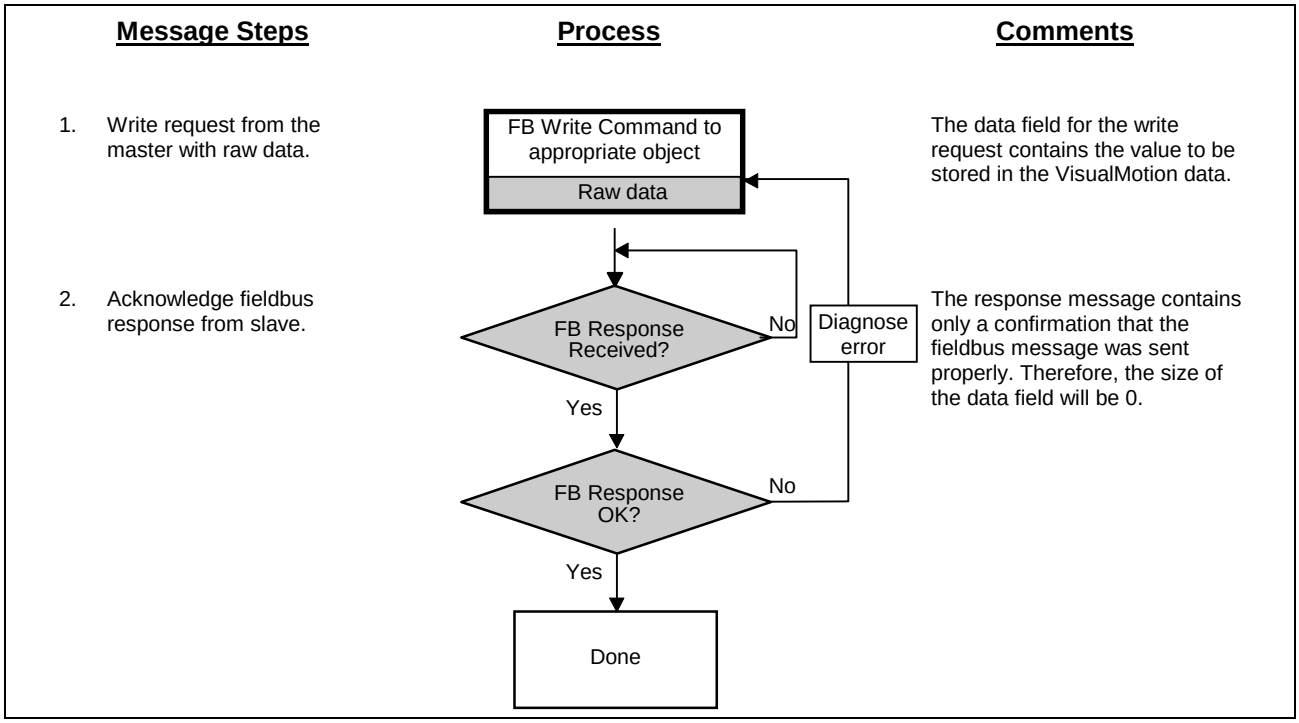
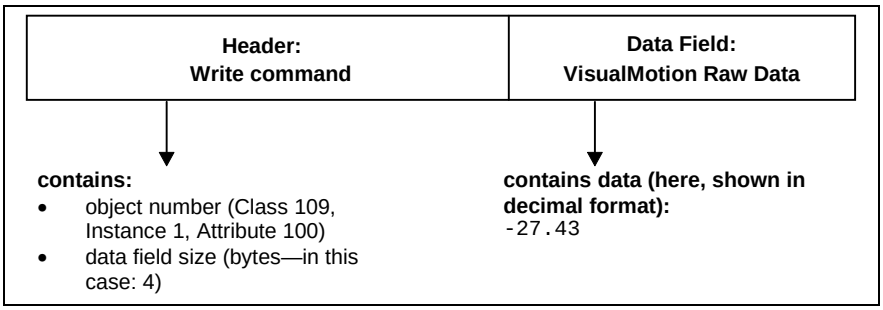

Message Steps
Process
Comments
1. Write request from the master with raw data.
2. Acknowledge fieldbus response from slave.
FB Write Command to appropriate object
Raw data
The data field for the write request contains the value to be stored in the VisualMotion data.
FB Response Received?
No
Diagnose error
Yes
The response message contains only a confirmation that the fieldbus message was sent properly. Therefore, the size of the data field will be 0.
FB Response OK?
No
Yes
Done

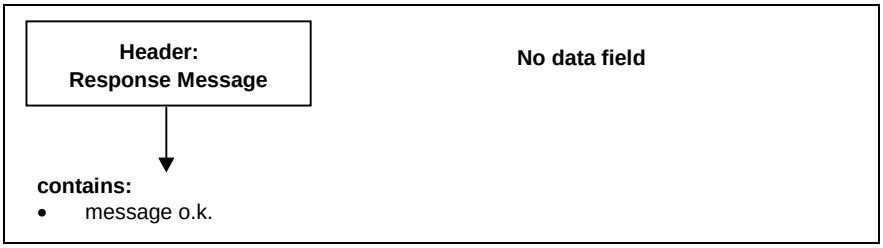
Figure 4-4: Non-Cyclic Direct-Mapped Write Process

Example: Write the value -27.43 to Float 16 (This is a 32-bit non-cyclic output object. In our Fieldbus Mapper example, it is mapped to object Class 109, Instance 1, Attribute 100.)

1. Write request from the master with raw data.



2. After the write request from the master, the CLC-D card sends a response message.



3. If the message response (code in message header) shows o.k., the transaction is complete.

Note: It is important to note the mapping of the object in the Fieldbus Mapper. If a particular object is mapped as a “VisualMotion Read Object,” you **cannot write** to that object. Conversely, if the object is mapped as a "VisualMotion Write Object," you **cannot read** from that object.

Non-Cyclic Direct-Mapped Read

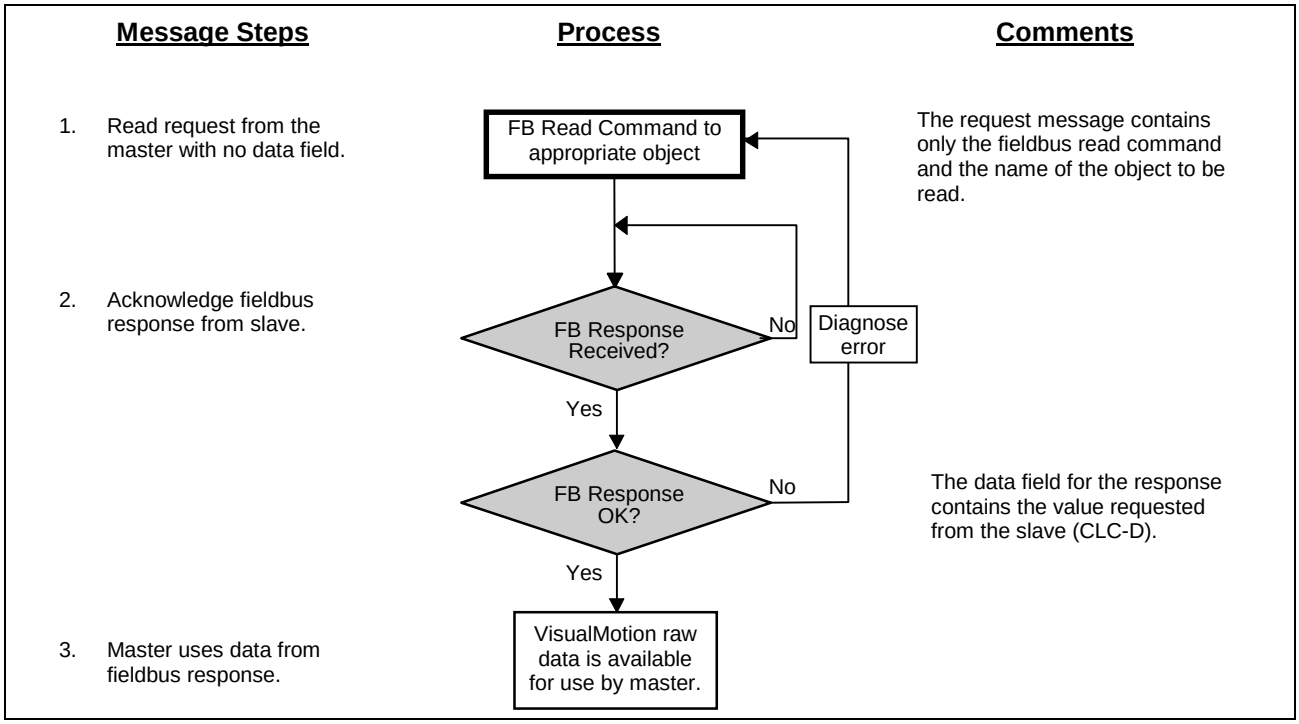
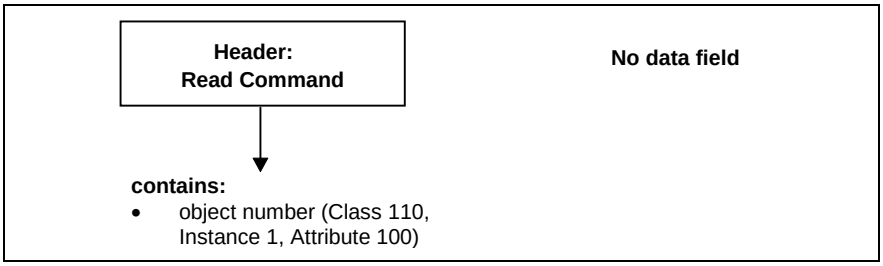

Message Steps
Process
Comments
1. Read request from the master with no data field.
2. Acknowledge fieldbus response from slave.
3. Master uses data from fieldbus response.
FB Read Command to appropriate object
FB Response Received?
No
Diagnose error
Yes
FB Response OK?
No
Yes
VisualMotion raw data is available for use by master.
The request message contains only the fieldbus read command and the name of the object to be read.
The data field for the response contains the value requested from the slave (CLC-D).

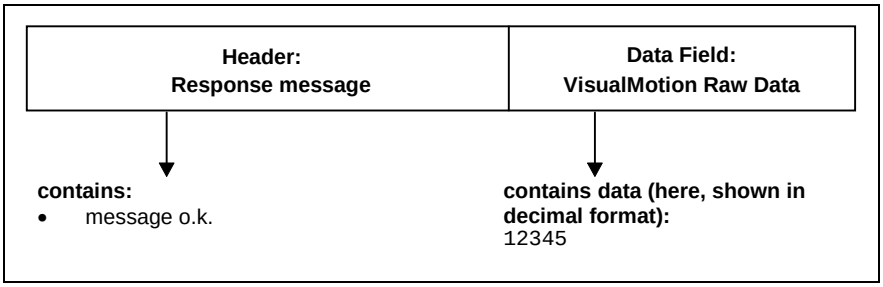
Figure 4-5: Non-Cyclic Direct-Mapped Read Process

Example: Read the value contained in Integer 8. (This is a 32-bit non-cyclic input object. In our Fieldbus Mapper example, it is mapped to object Class 110, Instance 1, Attribute 100.)

1. Read request from the master.



2. After the read request from the master, the CLC-D card sends a response message.



If the message response (code in message header) shows o.k., the requested value is attached to the message in the data field. This value is now available for use by the master (PLC).

Note: It is important to note the mapping of the object in the Fieldbus Mapper. If a particular object is mapped as a “VisualMotion Read Object,” you **cannot write** to that object. Conversely, if the object is mapped as a "VisualMotion Write Object," you **cannot read** from that object.

4.5 Non-Cyclic Transmission (Data Exchange Objects)

The four data exchange objects Class 100, Instance 1-4, Attribute 100 represent fixed data "containers" of varying lengths that transfer the VisualMotion ASCII Protocol to the CLC-D card. These objects serve as an open-ended possibility to access any VisualMotion data (including cams, diagnostic text, etc.), but more work is required in the master to perform a transmission of this type. Both the VisualMotion ASCII message and the Fieldbus transfer message must be formulated.

Table 4-1: Length of the Data Exchange Objects lists the available data exchange objects and their sizes.

Data Exchange Object	Data Length (in bytes)
Class 100 Instance 1 Attribute 100	16
Class 100 Instance 2 Attribute 100	32
Class 100 Instance 3 Attribute 100	64
Class 100 Instance 4 Attribute 100	128

Table 4-1: Length of the Data Exchange Objects

Selecting a Data Exchange Object

Depending on the length of a VisualMotion ASCII message, any of these data exchange objects can be selected.

Note: The entire data length of the data exchange object must always be transmitted even if the VisualMotion ASCII message is shorter.
For example, if you want to transmit an ASCII message of 42 bytes, you must use Class 100, Instance 3. To avoid a response error from the Fieldbus slave, you must append 22 "Null" characters to the end of the ASCII message to complete a data size of 64 bytes.

Note: The checksum for the VisualMotion ASCII protocol is NOT used with the data exchange object. If the checksum is sent as part of the string, it will be ignored, and no checksum will be sent in the VisualMotion ASCII response messages. To ensure data integrity, the Fieldbus protocols support a low-level checksum.

Transmission Sequence via a Data Exchange Object

Note: For the data exchange object, two transmissions (and two responses) are required, to send the read or write message to and then receive the response message from the CLC-D card.

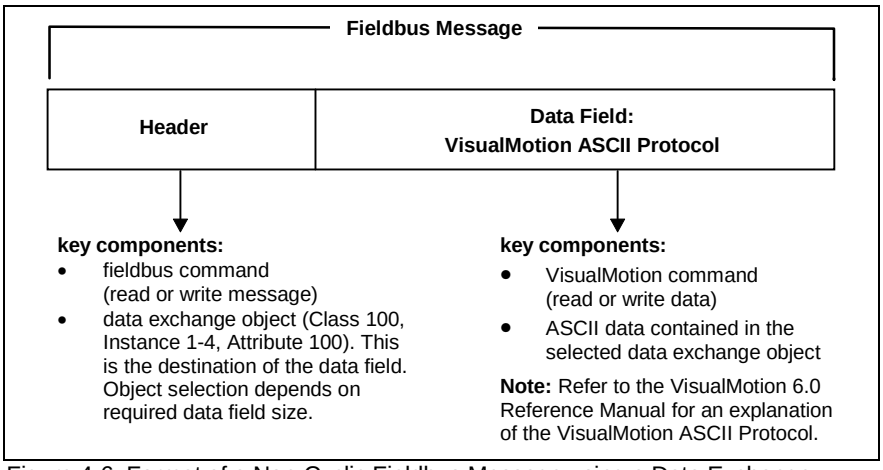

Fieldbus Message
Header
Data Field:
VisualMotion ASCII Protocol
key components:
• fieldbus command (read or write message)
• data exchange object (Class 100, Instance 1-4, Attribute 100). This is the destination of the data field. Object selection depends on required data field size.
key components:
• VisualMotion command (read or write data)
• ASCII data contained in the selected data exchange object
Note: Refer to the VisualMotion 6.0 Reference Manual for an explanation of the VisualMotion ASCII Protocol.

Figure 4-6: Format of a Non-Cyclic Fieldbus Message using a Data Exchange Object

Important: The format of the Fieldbus message header is dependent on the type of master (PLC) being used. Refer to your PLC manufacturer's manual for specific information on this topic.

The following sequence describes the communication between the Fieldbus master (PLC) and the Fieldbus slave (CLC-D):

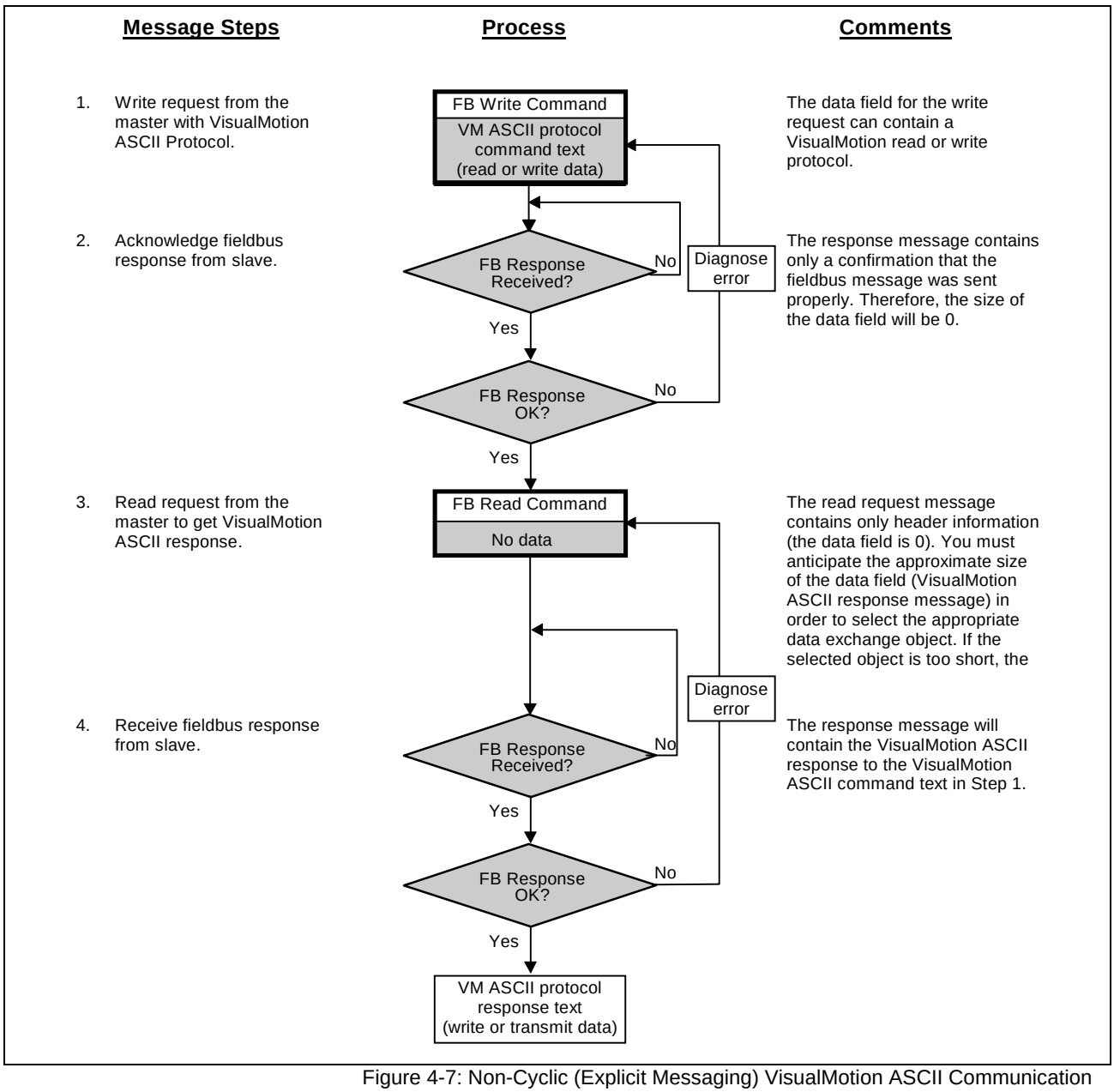
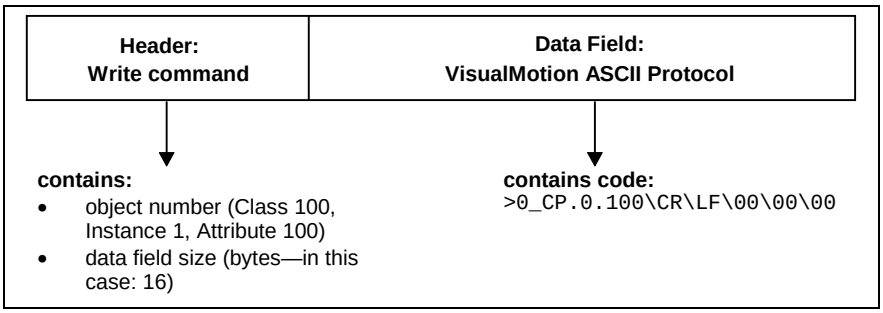

Message Steps
Process
Comments
1. Write request from the master with VisualMotion ASCII Protocol.
FB Write Command
VM ASCII protocol command text (read or write data)
The data field for the write request can contain a VisualMotion read or write protocol.
2. Acknowledge fieldbus response from slave.
FB Response Received?
No
Diagnose error
Yes
FB Response OK?
No
Yes
The response message contains only a confirmation that the fieldbus message was sent properly. Therefore, the size of the data field will be 0.
3. Read request from the master to get VisualMotion ASCII response.
FB Read Command
No data
The read request message contains only header information (the data field is 0). You must anticipate the approximate size of the data field (VisualMotion ASCII response message) in order to select the appropriate data exchange object. If the selected object is too short, the
4. Receive fieldbus response from slave.
Diagnose error
FB Response Received?
No
Yes
FB Response OK?
No
Yes
The response message will contain the VisualMotion ASCII response to the VisualMotion ASCII command text in Step 1.
VM ASCII protocol response text (write or transmit data)

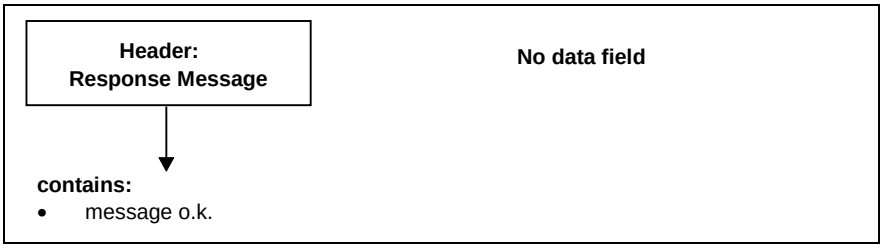
Figure 4-7: Non-Cyclic (Explicit Messaging) VisualMotion ASCII Communication Process

Example: Read Card Parameter 100 (CLC-D firmware version)

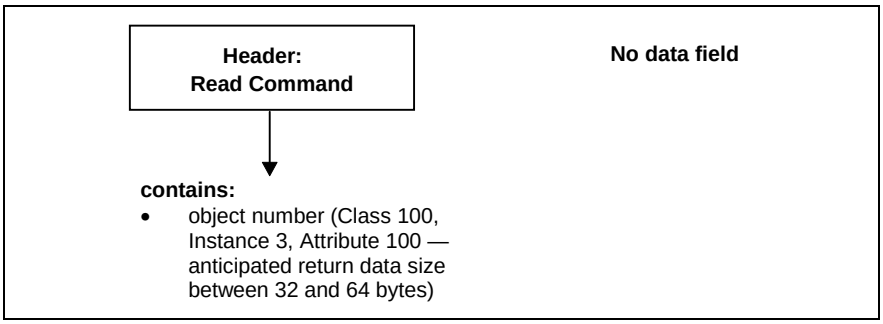
1. Write request from the master with VisualMotion ASCII Protocol.



2. After the first read request from the master, the CLC-D card sends a response message.

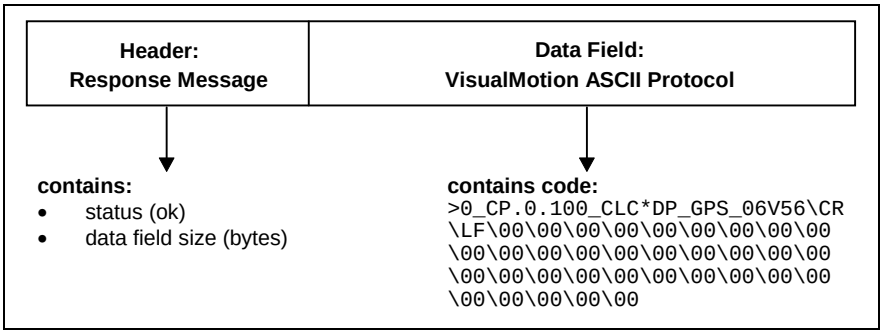


3. Read request from the master for the VisualMotion ASCII response message.


Header:
Read Command
No data field
contains:
• object number (Class 100, Instance 3, Attribute 100 — anticipated return data size between 32 and 64 bytes)

Note: To ensure that all of the data requested in this step is received in step 4 below, a data exchange object of the appropriate size must be selected.
If the selected data exchange object is too small, the data will be truncated.
If the selected data exchange object is too large, efficiency of transmission will be compromised.

4. The CLC-D sends the final response message.



5 DeviceNet DCF01.x Slave Board

5.1 Application

The DCF01.x DeviceNet Slave Card enables the inclusion of the CLC-D control card in a DeviceNet system. The DeviceNet card supports the real-time cyclic channel (Polled I/O) with up to 16 words and enables non-cyclic data exchange between master and slave via the non-cyclic Explicit Message DeviceNet channel.

With this hardware, CLC-D parameters and data sets, as well as parameters of an attached drive controller can be transmitted.

For the highest possible flexibility, the cyclic and non-cyclic channels are freely configurable by the user when using the object structure of the DCF01.x card.

5.2 Functionality

The DCF01.x card has the following characteristics:

- DeviceNet Slave according to ODVA specification 2.0.
- The card uses the predefined master/slave connection set and functions as a DeviceNet Group 2 Only server.
- DeviceNet startup with optically isolated interface. Connection via 5-pin Phoenix Combicon connector.
- Diagnostic 5-LED array on the front panel, according to ODVA guidelines.
- Freely configurable cyclic channel with 1-16 word capacity on the bus.
- Data transfer with the CLC-D control card via Dual-Port Memory.
- Hardware and software compatibility checking with the CLC-D control card.
- Implementation of an object structure to simplify access to variables and parameters in the control card and drives.
- Upload / Download function via 4 arrays, from 16 to 128 Bytes (Explicit Message Data Exchange Object).
- Via Fragmented Explicit Messaging, a maximum of 128 bytes of data can be transmitted (largest data exchange object).
- Available baud rates: 125, 250 and 500 kBps, selectable via a CLC-D parameter. Default value: 125 kBps.
- MAC-ID includes the slave's device address and is set via the CLC-D. Default value: 63.
- For DeviceNet scanners operating in Motorola data format, but not configurable for double-word and byte arrays, swapping can be set in the CLC-D Fieldbus Mapper.

5.3 DCF01.x Card Hardware

Front View of the DCF01.x

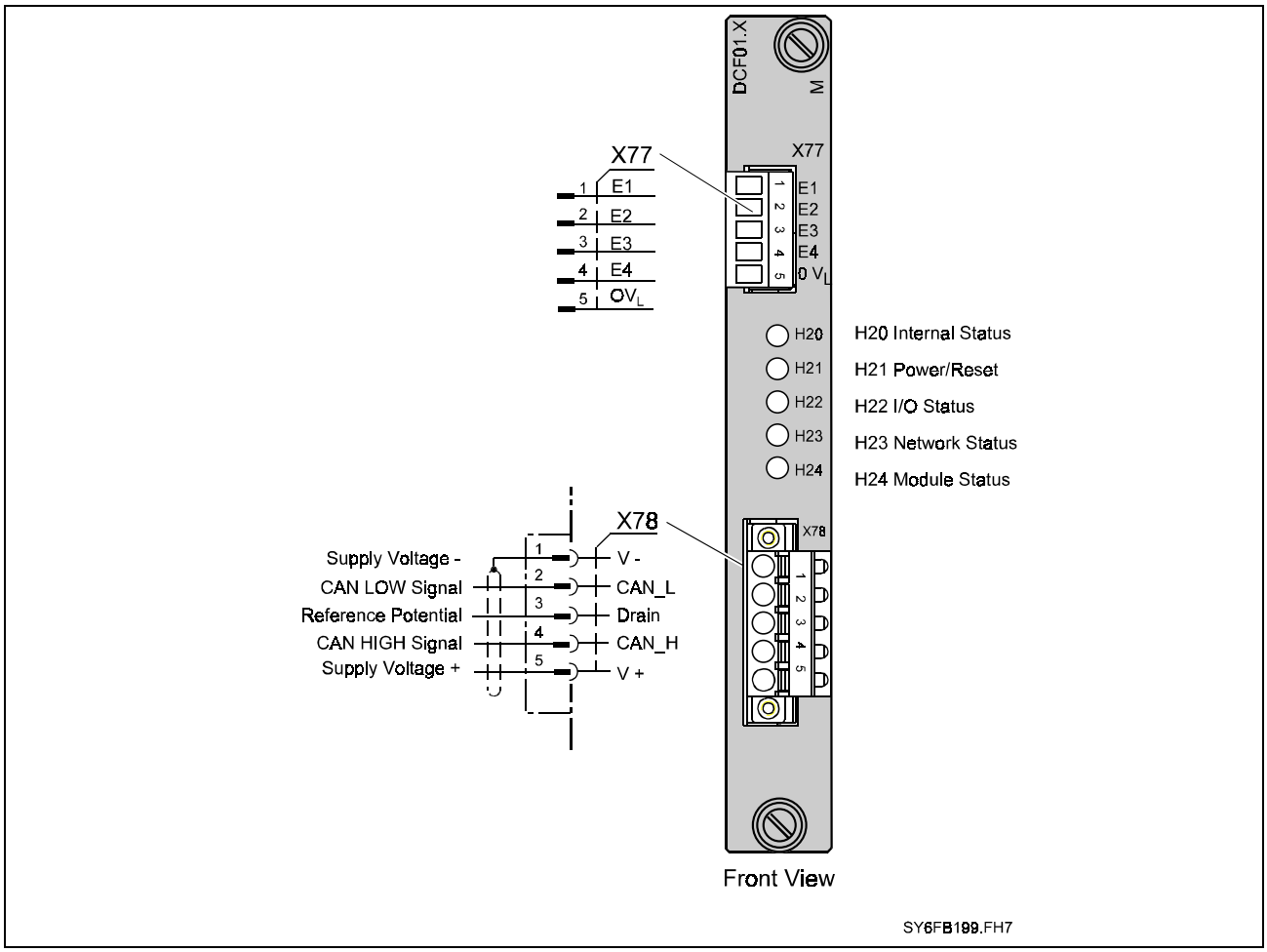


Figure 5-1: Front View of the DCF01.x

DCF01.x Structure

The DCF01.x card is a plug-in card that can be directly connected to the CLC-D controller card. It can be inserted into a drive or CCD card cage as one unit (occupying two card slots).

Note: Other cards can be plugged into the DCF01.x. Please take notice when disassembling or removing from the card cage.

Power (+5V) is supplied by the drive or CCD card cage via another connector on the back of the card. However, signals are always transmitted via the plug-in connection to the CLC-D.

The interface to the CLC-D is achieved via an Indramat-specific extended bus. However, only one DCF01.x card can be connected to the CLC-D.

Note: The DCF01.x card cannot be used with a DPF05.x Profibus slave card or a DBS03.x Interbus slave card.

The DCF01.x card has the following interfaces:

Interface to the Drive or CCD Card Cage

required only for power supply to the DCF01.x if power is not supplied by the CLC-D.

Interface to the CLC-D or other Cards

Information is transmitted to the CLC-D and any other cards via this interface. It is an extended bus interface with partial coding of the adressing field, so that additional cards can be connected.

External Inputs

The DCF01.x card provides three hardware inputs (+24V). These inputs can only be used with the CLC-D if the appropriate firmware is available. The signal status at these inputs are transmitted to the CLC-D independently of the DeviceNet Status (On/Off), but can also be requested from the DeviceNet Master via Polled I/O or Explicit Message (these external inputs are currently not supported).

External Power

The DCF01.x transceivers are supplied with power via the bus cable according to ODVA specifications. This does not mean that the entire unit is supplied via the bus. The CLC-D control operates only if it has a separate power supply.

DeviceNet Interface

The DeviceNet interface is designed according to ODVA specifications and is optically isolated. Connection can be made via a 5-pin Phoenix Combicon connector.

Connector Pinouts

X78 Connector, DeviceNet

X78	Signal	Description	Color
1	V-	Supply Voltage -	Black
2	CAN_L	CAN LOW Signal	Blue
3	Drain	Reference Potential	<blank>
4	CAN_H	CAN HIGH Signal	White
5	V+	Supply Voltage +	Red

Table 5-2: X78 Pinouts, DeviceNet

X77 Pinouts, External Inputs

(not implemented at this time)

X77	Description	Input Voltage, High	Input Voltage, Low
1	E1	+16 V... +32 V	-0.5 V ... +8 V
2	E2	+16 V... +32 V	-0.5 V ... +8 V
3	E3	+16 V... +32 V	-0.5 V ... +8 V
4	E4	+16 V... +32 V	-0.5 V ... +8 V
5	$0V_L$	Reference Potential 0V	Reference Potential 0V

Table 5-3: X77 Pinouts, External Inputs

DCF01.x Diagnostics

Front Panel LEDs

The DCF01.x card has 5 diagnostic LEDs on the front panel. They allow diagnosis of the status of the bus commincation and the communication between the DCF01.x and CLC-D cards.

Definition of the Diagnostic LEDs

LED	Signal	Status	Definition
H20	**Internal Status**	Red	No synchronization with CLC-D
		Off	Successful synchronization with CLC-D card
		Green (intermittent)	Explicit Message received
H21	**Power/Reset**	Off	Unit off or Reset signal active
		Green	Backplane (**not** Transceiver) Supply Voltage OK
H22/IOS	**I/O Status**	Off	All Inputs and Outputs inactive
		Green	Outputs active and Inputs valid
		Green (flashing)	Outputs inactive (no longer being sent by master)
		Red (flashing)	Limit exceeded for I/O connection (timeout), but Inputs still active
H23/NS	**Network Status**	Off	Not on-line
		Green (flashing)	On-line, but no connection to Master
		Green	On-line, with connection to Master
		Red (flashing)	Timeout limit exceeded for I/O connection
		Red	Critical connection error (duplicate MAC ID or Bus off)
H24/MS	**Module Status**	Off	Unit off
		Green	Normal operation
		Green (flashing)	Configuration Error
		Red (flashing)	Recoverable Error
		Red	Unrecoverable Error, replace Card

Table 5-4: LED Definitions

5.4 DeviceNet Polled I/O (Cyclic Channel)

DCF01.x Polled I/O Configuration

The DCF01.x card is an intelligent DeviceNet card that can be configured according to process requirements. Not only does it provide a particular number of words which are transmitted in both data directions during each DeviceNet cycle, but it also allows the user to determine which data word is placed in which location in the data segment.

This flexibility enables assigning data to adressable elements, or objects, that are designated with index numbers for different Fieldbuses. The user can then choose the required objects from the configuration stored on the CLC-D board, and place them in the desired order in the Polled I/O.

Each time the DCF01.x DeviceNet card is powered up with the CLC-D card, the CLC-D configuration is transmitted.

User-Specific Polled I/O Configuration

The user can determine any desired configuration of the Polled I/O via the CLC-D configuration.

Note: Up to 16 words can be configured on the bus when using the DCF01.x card.

Note: Changes to word length always lead to the reconfiguration of the bus. This is valid for both process input and output data, and is signaled by a CRC error initiated by the slave. This error requires reconfiguration via the master.

Monitoring the DCF01.x Polled I/O

Data transfer on the DeviceNet is generally categorized as secure. If data is transferred from the master to the slaves, it is subjected to extensive CRC checks, before it is accepted as valid data and activated by the slave. Data which is recognized as erroneous (HD = 6, for DeviceNet), is not forwarded to the CLC-D. Of course, this is also true for the data direction to the master.

Watchdog Function: The DCF01.x card provides bus monitoring by using a watchdog timer, so that an appropriate reaction can be effected when the bus is down.

For Class 5, Instance 2, Attribute 9 (expected packet rate) the time in ms is derived from the linking object by multiplying by 4, according to the ODVA specification. As an error reaction, bit 3 of the Fieldbus status word is reset to 5FF2. See ***Register 19 Definition (Fieldbus Status)*** on page 3-6.

Non-Cyclic Objects

The DeviceNet implementation for CLC-D is based on previous Fieldbus implementations (Interbus and Profibus). Although DeviceNet data objects are expressed according to Class, Instance and Attribute, there is a correlation between these objects and the objects used in Interbus and Profibus.

Non-Cyclic, Data Exchange Objects

The following are supported:

Object	Class	Instance	Attribute
16-byte	100	1	100 (Data) 101 (Length in bytes) 102 (Indramat Object ID, read only)
32-byte	100	2	100 (Data) 101 (Length in bytes) 102 (Indramat Object ID, read only)
64-byte	100	3	100 (Data) 101 (Length in bytes) 102 (Indramat Object ID, read only)
128-byte	100	4	100 (Data) 101 (Length in bytes) 102 (Indramat Object ID, read only)

Non-Cyclic, Direct-Mapped Objects

The following are supported:

Object	Class	Instance	Attribute
32-bit, input	110	1-16	100 (Data) 101 (Length in bytes) 102 (Indramat Object ID, read only)
32-bit, output	110	1-16	100 (Data) 101 (Length in bytes) 102 (Indramat Object ID, read only)
16-bit, input	115 and 119	1-16	100 (Data) 101 (Length in bytes) 102 (Indramat Object ID, read only)
16-bit, output	116 and 120	1-16	100 (Data) 101 (Length in bytes) 102 (Indramat Object ID, read only)

6 Index

*

*.gsd file....4-1
Basic Example 2-3
Multiplex Example 2-14

2

208 Lost Fieldbus Connection..3-10
209 Field Bus Mapper Timeout.3-10

5

519 Lost Fieldbus Connection..3-10
520 Field Bus Mapper Timeout.3-10
5E70.... 4-19
5E71.... 4-19
5E72.... 4-19
5E73.... 4-19

A

Available Objects
Basic Example 2-4
Multiplex Example
Cyclic Data....2-14

B

baud rates
supported by DPF05.x.... 5-1
Bit Definitions
Register 19
Cyclic Data Valid....3-7
FB Init OK 3-6
FB Slave Ready 3-6
nCyc Chan Ready 3-7
Non-Cyc Ready....3-7
nVM FW OK....3-7
Register 20
FB Card Fault Code 3-8
FB Card Found 3-8
Register 26
Current Miss Counter 3-9
Fieldbus Timeout Counter....3-9
Peak Miss Counter....3-9
Bus Conf. IN List
Basic Example 2-3
Multiplex Example
Cyclic Data....2-14
Bus Conf. OUT List
Basic Example 2-5
Multiplex Example
Cyclic Data 2-15
Bus Configuration Lists
APPEND Button 2-4, 2-14
DEL Button 2-4, 2-14
INSERT Button.... 2-4, 2-14
NEW Button 2-4, 2-14
Bus Type
Basic Example.... 2-2
Cyclic Data 2-6
Multiplex Example
Cyclic Data 2-12, 2-16
Non-Cyclic Data.... 2-18
Buttons
APPEND 2-4, 2-14
DEL 2-4, 2-14
Display Summary 2-7, 2-9, 2-18, 2-19
Exit....2-7, 2-9, 2-18, 2-19
Get From CLC.....2-7, 2-9, 2-18, 2-19
Get From File 2-7, 2-9, 2-18, 2-19
INSERT 2-4, 2-14
NEW..... 2-4, 2-7, 2-9, 2-14, 2-18, 2-19
Save To File 2-7, 2-9, 2-18, 2-19
Send To CLC.....2-7, 2-9, 2-18, 2-19

C

C-0-2600..... 2-7, 2-9, 2-18, 2-19
C-0-2600....1-2
C-0-2635....3-9
C-0-2700..... 1-7, 2-9, 2-18, 2-19
channels
Cyclic Data Channel configuration1-3, 1-6
multiplex 1-2
Non-Cyclic Channel.... 1-2
Parameter Channel 1-2
Real-Time Channel 1-2, 1-3
single 1-2
Clearing Parameter Channel Errors4-4
Command Telegram Structure
Short Format 2 4-7
VisualMotion ASCII Format.... 4-11
Configure Multiplex Channel
Basic Example.... 2-3
control word 1-5
Multiplex Example
Cyclic Data 2-14
Cyclic Channel....1-2
Cyclic Data Channel
Basic Example.... 2-6
configuration.... 1-3, 1-6

Multiplex Example 2-16
Cyclic Data Valid 3-7

D

Data Configuration List
Param..... 2-20
Data Exchange Objects...**1-8**, 4-19
5E70 4-19
5E70 to 5E73 1-8
5E71 4-19
5E72 4-19
5E73 4-19
Data Mapping
Direct-Mapped Data **1-7**
Data Objects.....**1-7**
Data Sizes
Cyclic..... 1-4
Data Type
Basic Example
Cyclic Data.....2-6
Non-Cyclic Data 2-8
Multiplex Example
Cyclic Data.....2-16
Non-Cyclic Data 2-18
Data Types
Cyclic..... 1-4
Multiplex 1-3, 1-4, 2-11
Single 1-3, 1-4
Define/Config Slave Objects
Basic Example 2-3
Multiplex Example
Cyclic Data.....2-13
Destination
Basic Example
Cyclic Data.....2-6
Non-Cyclic Data.....2-8
Multiplex Example
Cyclic Data.....2-16
Non-Cyclic Data 2-18
Device Address
Basic Example.....2-3
Multiplex Example
Cyclic Data.....2-13
Diagnostics
LEDs 5-3
Direct-Mapped Data.....**1-7**
Display Summary
Cyclic, Multiplex Data..... 2-18
Cyclic, Single Data..... 2-7
File Menu entry..... 3-1
Non-Cyclic, Multiplex Data 2-19
Non-Cyclic, Single Data 2-9
DPF05.x
Hardware..... 5-2
Interfaces..... 5-2
DPF05.x board.....1-1

E

EMC safety 5-1
Error Bit (E).....**4-4**
Error Reaction
Ignore 3-10
Shutdown CLC 3-10
Warning Only..... 3-10
Errors
208 Lost Fieldbus Connection..... 3-10
209 Field Bus Mapper Timeout 3-10
519 Lost Fieldbus Connection..... 3-10
520 Field Bus Mapper Timeout 3-10
Exit
Cyclic, Multiplex Data 2-18
Cyclic, Single Data 2-7
Non-Cyclic, Multiplex Data 2-19
Non-Cyclic, Single Data 2-9
External Inputs.....5-3

F

FB Card Fault Code.....3-8
FB Card Found 3-8
FB Init OK 3-6
FB Slave Ready.....3-6
Field Bus Error Reaction
Basic Example..... 2-3
Multiplex Example
Cyclic Data 2-14
Field Bus Mapper
Examples
Multiplexing..... 2-11
Parameter Channel..... 2-20
Examples
Basic 2-1
Fieldbus Diagnostics.....3-7
Fieldbus Error Reaction 3-9
Fieldbus Mapper 1-1
Fieldbus Master 1-1
fieldbus message header.....4-16
Fieldbus Resource Monitor.....3-8
Fieldbus Slave 1-1
Fieldbus Status 3-6
fieldbus summary report 2-7
Fieldbus-Accessible Parameters 3-3
File Menu
Display Summary 3-1

Print ... 3-2
first cycle ... **4-4**
FMS (Fieldbus Message Specification) ... 1-2
From FieldBus
Basic Example ... 2-4
Multiplex Example
Cyclic Data ... 2-15

G

Get From CLC
Cyclic, Multiplex Data ... 2-18
Cyclic, Single Data ... 2-7
Non-Cyclic, Multiplex Data ... 2-19
Non-Cyclic, Single Data ... 2-9
Get From File
Cyclic, Multiplex Data ... 2-18
Cyclic, Single Data ... 2-7
Non-Cyclic, Multiplex Data ... 2-19
Non-Cyclic, Single Data ... 2-9
GPS Programming ... 3-1

H

H15A ... 5-4
H15B ... 5-4
H15C ... 5-4
H15D ... 5-4
H16A ... 5-4
H16B ... 5-4
H16C ... 5-4

I

Ignore (Fieldbus Error Reaction) 3-10
Index Number
Multiplex Example
Cyclic Data ... 2-14
input ... 1-2

L

last cycle ... **4-4**
least significant bit ... 3-6
LEDs
H15A ... 5-4
H15B ... 5-4
H15C ... 5-4
H15D ... 5-4
H16A ... 5-4
H16B ... 5-4
H16C ... 5-4
LEDs (DPF05.x) ... 5-3
Length of DP Channel (Word)
Basic Example ... 2-3
Multiplex Example
Cyclic Data ... 2-13

M

Mapping Direction
Basic Example
Cyclic Data ... 2-6
Non-Cyclic Data ... 2-8
Multiplex Example
Cyclic Data ... 2-16
Non-Cyclic Data ... 2-18
Mapping List
Back To Add Button ... 2-8
Delete Button ... 2-8
Insert Button ... 2-8
Replace Button ... 2-8
Messaging Examples
Short Format 2 ... **4-8**, 4-13
Messaging Formats
Short Format 2 ... **1-6**
VisualMotion ASCII Format .. **1-6**, **4-11**
most significant bit ... 3-6
Multiplex Data Bits ... ***4-1***
Multiplex Data Types . 1-3, 1-4, 2-11
Multiplexing
Control and Status Words ... **1-5**
Enable ... 2-14

N

nCyc Chan Ready ... 3-7
New
Cyclic, Multiplex Data ... 2-18
Cyclic, Single data ... 2-7
Non-Cyclic, Multiplex Data ... 2-19
Non-Cyclic, Single Data ... 2-9
Non-Cyc Ready ... 3-7
Non-Cyclic Channel ... 1-2, **1-7**
Non-Cyclic Data Channel
Multiplex Example ... 2-18
Non-Cyclic Data Mapping Lists
Direct Mapping
Basic Example ... 2-8
Multiplex Example ... **2-18**
Non-Cyclic Transmission
Data Exchange Objects ... 4-19
Number of Valid Data Bytes (#Bytes) ... **4-5**
nVM FW OK ... 3-7

O

OBJECT MAPPING LIST
- Basic Example
  - Cyclic Data....2-6
  - Non-Cyclic Data....2-9
- Multiplex Example
  - Cyclic Data....2-17
  - Non-Cyclic Data....2-18

OK, Send to CLC
- Basic Example....2-5
- Multiplex Example
  - Cyclic Data....2-15

output....1-2

P

Parameter Channel ... 1-2, 1-6, 2-20
- Short Format 2....1-6
- VisualMotion ASCII Format....1-6

Parameter Mode
- Basic Example....2-7
- Multiplex Example....2-18

Parameters....3-3

P-CHAN Components
- Control/Status Word....4-4
- **Error Bit (E)....4-4**
- **first cycle....4-4**
- **last cycle....4-4**
- **Number of Valid Data Bytes (#Bytes)4-5**
- **Toggle Bit (T)....4-5**
- **Transmission Format (Fmt)....4-5**

Pinouts (DPF05.x)....5-3

PLC Programming....4-1
- ***Multiplex Data Bits....4-1***
- Non-Cyclic Data (via Data Exchange Objects)....4-19
- Non-Cyclic Data (via Direct Mapping)4-16
- Parameter Channel....4-3

Print
- File Menu entry....3-2

Print a Summary Report....3-2

Profibus DP Combi Slave Board.5-1

R

Real-Time Channel....1-2, 1-3

Register 19 Definition (Fieldbus Status)....3-6

Register 20 Definition (Fieldbus Diagnostics)....3-7

Register 26 Definition (Fieldbus Resource Monitor)....3-8

Response Telegram Structure
- Short Format 2....4-7
- VisualMotion ASCII Format....4-12

S

Save to File
- Cyclic, Multiplex Data....2-18
- Cyclic, Single Data....2-7
- Non-Cyclic, Multiplex Data....2-19
- Non-Cyclic, Single Data....2-9

Send To CLC
- Cyclic, Multiplex Data....2-18
- Cyclic, Single Data....2-7
- Non-Cyclic, Multiplex Data....2-19
- Non-Cyclic, Single Data....2-9

Send to CLC
- Basic Example
  - Cyclic Data....2-7
  - Non-Cyclic Data....2-9
- Multiplex Example
  - Cyclic Data....2-17

Short Format 2....1-6
- Command Telegram Structure....4-7
- Messaging Example....4-13
- **Messaging Examples....4-8**
- Response Telegram Structure....4-7

Shutdown CLC (Fieldbus Error Reaction)....3-10

Single Data Types....1-3, 1-4

Source Object list
- Basic Example
  - Non-Cyclic Data....2-9

status word....1-5
- Multiplex Example
  - Cyclic Data....2-14

T

To FieldBus
- Basic Example....2-3
- Multiplex Example
  - Cyclic Data....2-14

Toggle Bit (T)....4-5

Transmission Format (Fmt)....4-5

V

View a Summary Report....3-1

VisualMotion ASCII Format1-6, **4-11**
- Command Telegram Structure....4-11
- Response Telegram Structure....4-12

W

Warning Only (Fieldbus Error Reaction)....3-10

X

X68 Connector.....5-3
X69 Connector.....5-3

engineering
mannesmann
Rexroth

Supplement C
Interbus Fieldbus Interfaces
VisualMotion 6.0

Indramat

Contents

1 General Information 1-1
1.1 CLC-D System Description with a Fieldbus 1-1
The VisualMotion Fieldbus Mapper 1-1
1.2 Data Transfer Direction (Output vs. Input) 1-2
1.3 Fieldbus Data Channel Descriptions 1-2
Cyclic Channel 1-2
The Real-Time Channel 1-4
Non-Cyclic Channel 1-6
Direct-Mapped Data 1-6
Data Exchange Objects 1-7

2 FieldBus Mapper Examples 2-1
2.1 Basic Example 2-1
STEP I: Determine the Cyclic and Non-Cyclic Data 2-1
STEP II: Configure Fieldbus Slave Card with Setup Information and Cyclic Data Object Lists Using the Fieldbus Mapper 2-2
STEP III: Define Cyclic Data Mapping Lists 2-5
STEP IV: Define Non-Cyclic Data Mapping Lists (Direct Mapping) 2-8
2.2 Multiplexing Example 2-10
STEP I: Determining the Cyclic and Non-Cyclic Data (with Multiplexing) 2-10
STEP II: Configure Fieldbus Slave Card with Setup Information and Cyclic Data Object Lists Using the Fieldbus Mapper 2-11
STEP III: Define Cyclic Data Mapping Lists 2-15
STEP IV: Define Non-Cyclic Data Mapping Lists (Direct Mapping) 2-17

3 Information for the GPS Programmer 3-1
3.1 Viewing/Printing a Summary Report of the Current Fieldbus (only when CLC-D is on-line) 3-1
To View a Summary Report: 3-1
To Print a Summary Report: 3-2
3.2 Fieldbus-Accessible Parameters 3-3
3.3 Register 19 Definition (Fieldbus Status) 3-6
Diagnostic Object 5FF2 3-6
Bit Definitions 3-6
3.4 Register 20 Definition (Fieldbus Diagnostics) 3-7
Diagnostic Object 5FF0 3-7
Bit Definitions 3-8
3.5 Register 26 Definition (Fieldbus Resource Monitor) 3-8
Bit Definitions 3-9
3.6 Fieldbus Error Reaction 3-9

4 Information for the PLC Programmer 4-1
4.1 Slave Configuration 4-1
4.2 Multiplex Data Bits in the Control and Status Words 4-1
4.3 Non-Cyclic Transmission (Direct-Mapped Objects) 4-3
Selecting a Direct-Mapped Object 4-3
Transmission Sequence via a Direct-Mapped Object 4-3
Non-Cyclic Direct-Mapped Write 4-4
Non-Cyclic Direct-Mapped Read 4-5
4.4 Non-Cyclic Transmission (Data Exchange Objects) 4-6
Selecting a Data Exchange Object 4-6
Transmission Sequence via a Data Exchange Object 4-6

5 Interbus Slave Board DBS03.x 5-1
5.1 Application 5-1
5.2 Functionality 5-1
5.3 Bus Configuration 5-1
5.4 DBS03.x Board Hardware 5-2
Front view of the DBS03.x 5-2
DBS03.x Structure 5-2
X40 Connector, Interbus Pin-outs, Incoming Bus 5-3
X41 Connector, Interbus Pin-outs, Outgoing Bus 5-3
X39 Connector, External Inputs 5-4
DBS03.x Diagnostics 5-4
Front Panel LEDs 5-4
Definition of Diagnostic LEDs 5-4
5.5 Interbus Process Data (Cyclic Channel) 5-5
DBS03.x Process Data Configuration 5-5
Monitoring the DBS03.x Process Data 5-5
5.6 Interbus Peripherals Communication Protocol (Non-Cyclic Channel) 5-6
Supported PCP Services for the DBS03.x 5-6
DBS03.x PCP Firmware 5-6
Addressing of PCP Devices on the Bus 5-7
5.7 DBS03.x Objects 5-7
Objects for Diagnosing and Configuring the DBS03.x 5-7
Diagnostic Array 5E7A 5-8

6 Index 6-1

1 General Information

Important: Using a Fieldbus with the VisualMotion system will consume additional processing resources on the CLC-D card. Please take into consideration the consumption of these resources, especially when adding a Fieldbus to an existing application.

Version Note:
Information in this document is based on VisualMotion Toolkit software version 06V12 and CLC-D firmware version GPS06V61.

1.1 CLC-D System Description with a Fieldbus

The CLC-D can operate on a serial Fieldbus interface (network) by means of a plug-in card (DBS03.x board) that communicates with the CLC-D card via dual-port RAM. The function of the Fieldbus card is similar to that of a network card in a PC: it allows communication with other devices on the network.

In ***Figure 1-1: Sample Master/Slave Setup with Fieldbus Cards***, a commonly described Fieldbus interface is pictured:

- **Fieldbus Master** - PLC Fieldbus interface
- **Fieldbus Slave** - CLC-D Fieldbus interface

In this document, we will refer to the PLC as the **Fieldbus master** and the CLC-D as the **Fieldbus slave**.

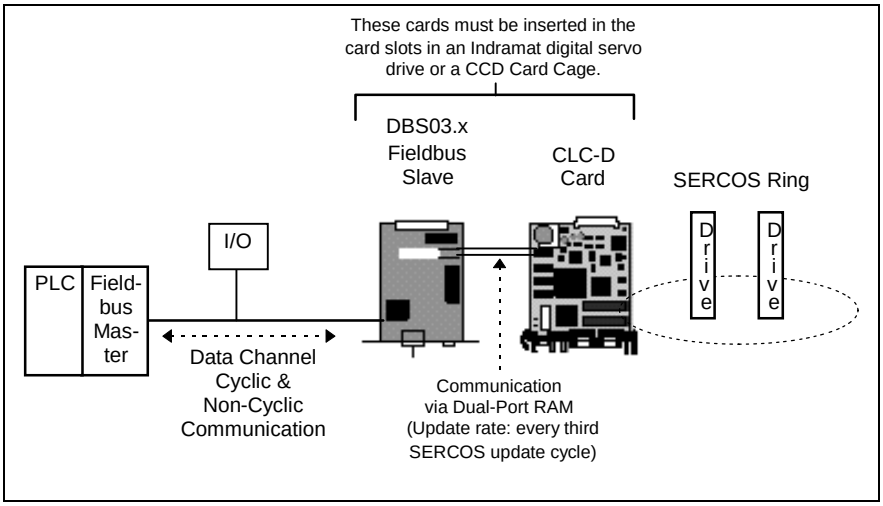

These cards must be inserted in the card slots in an Indramat digital servo drive or a CCD Card Cage.
DBS03.x Fieldbus Slave
CLC-D Card
SERCOS Ring
I/O
PLC
Field-bus Mas-ter
Drive
Drive
Data Channel Cyclic & Non-Cyclic Communication
Communication via Dual-Port RAM (Update rate: every third SERCOS update cycle)

Figure 1-1: Sample Master/Slave Setup with Fieldbus Cards

With the CLC-D card, the Interbus (DBS03.x) Fieldbus card can be used **only** as a **slave** card in a master/slave setup.

Important: When using a Fieldbus slave interface, it is recommended to use a 4 ms or higher SERCOS update.

The VisualMotion Fieldbus Mapper

In the VisualMotion software package, the Fieldbus Mapper is a tool used to set up Fieldbus data. It is primarily an on-line tool, used most effectively when connected to a CLC-D card.

Important: Whenever the Fieldbus Mapper is configured on-line, the system must be in Parameter Mode.

1.2 Data Transfer Direction (Output vs. Input)

In the VisualMotion Fieldbus Mapper, output and input are always described with respect to the Fieldbus master (the Fieldbus card associated with the PLC). The definitions for output and input follow:

output: the communication from the PLC to the CLC-D card (i.e. from the Fieldbus master to the Fieldbus slave).
Synonyms for this type of communication: **send** or **write** data.

input: the communication from the CLC-D card to the PLC (i.e. from the Fieldbus slave to the Fieldbus master).
Synonyms for this type of communication: **receive** or **read** data.

1.3 Fieldbus Data Channel Descriptions

The Indramat Interbus Fieldbus interface card for the CLC-D (DBS card) supports two data channels for input/output communication with the CLC-D:

1. **Cyclic Channel: PD** (Process Data)
   Real-Time Channel (for single and multiplex channels)
2. **Non-Cyclic Channel:** PCP (Peripheral Communications)

Cyclic Channel

The DBS03.x Fieldbus card has pre-defined cyclic bus objects (16- or 32-bit) which are declared and transmitted as an ordered list (the bus configuration list). This bus configuration list acts as a placeholder for CLC-D mapped data from the CLC-D object mapping list. See *Object Lists and Their Transfer Locations.*

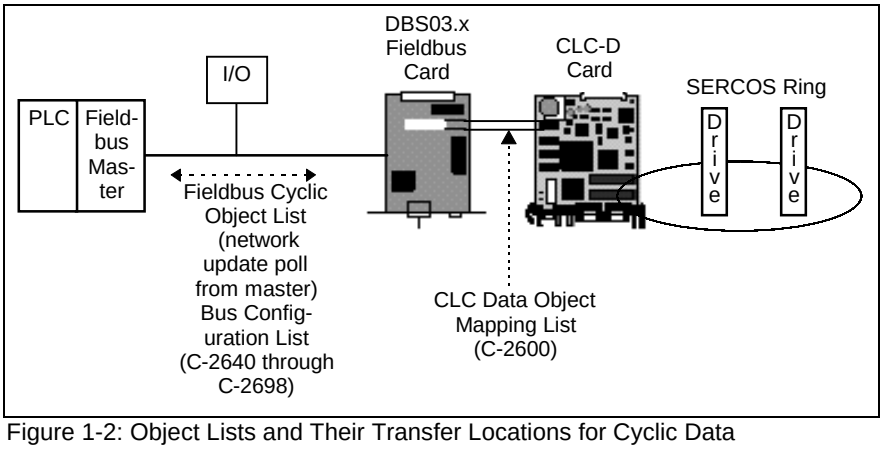


Figure 1-2: Object Lists and Their Transfer Locations for Cyclic Data

The cyclic data channel is limited to a total of 15 input words and 15 output words (with one additional data word being consumed by the non-cyclic [PCP] channel at all times). CLC-D data types consume these objects in either one-word (or 16-bit) groups for CLC registers or two-word (or 32-bit) groups for all other data types in the object mapping list (C-0-2600).

According to the Interbus specification for valid data transfer lengths, not every increment (number of data words) is valid. See the following table for the valid Interbus channel lengths when multiplexing is off:

Valid Interbus Channel Lengths	=	**Number of Configurable Cyclic Data Words** ****NOTE:** This is the number that must be entered in the Field Bus Mapper Screen under "Length of PD Channel (word)."	+	**Words Consumed by Non-Cyclic Channel (PCP)**	+	**Multiplexing Off - Words Consumed by Control/Status Word**
1-10		1-9		1		0
12		11		1		0
14		13		1		0
16		15		1		0

Table 1-1: Valid Interbus Channel Lengths (Multiplexing OFF)

When using multiplexing, the valid lengths of the cyclic data channel are reduced by one because of the control/status word, which always consumes the last used word of the PD channel (both for input and output directions) when multiplexing is on. See the table below for the valid Interbus channel lengths when multiplexing is on:

Valid Interbus Channel Lengths	=	Number of Configurable Cyclic Data Words ****NOTE:** This is the number that must be entered in the Field Bus Mapper Screen under "Length of PD Channel (word)."	+	Words Consumed by Non-Cyclic Channel (PCP)	+	Multiplexing Off - Words Consumed by Control/Status Word
1-10		1-8		1		1
12		10		1		1
14		12		1		1
16		14		1		1

Table 1-2: Valid Interbus Channel Lengths (Multiplexing ON)

Note: The bus configuration list must be set up in the Fieldbus mapper bus configuration screen before the object mapping list can be configured.

For the cyclic data, the CLC-D data object mapping list is scanned every third SERCOS update cycle and data is sent and received to/from the slave board's dual port RAM.

The cyclic data channel can be made up of any combination of the following data types:

- Real-Time Channel
  - Single Channel
  - Multiplex Channel
- Non-Cyclic Data from the PCP Channel (1 word fixed to the end of the real-time channel at all times)

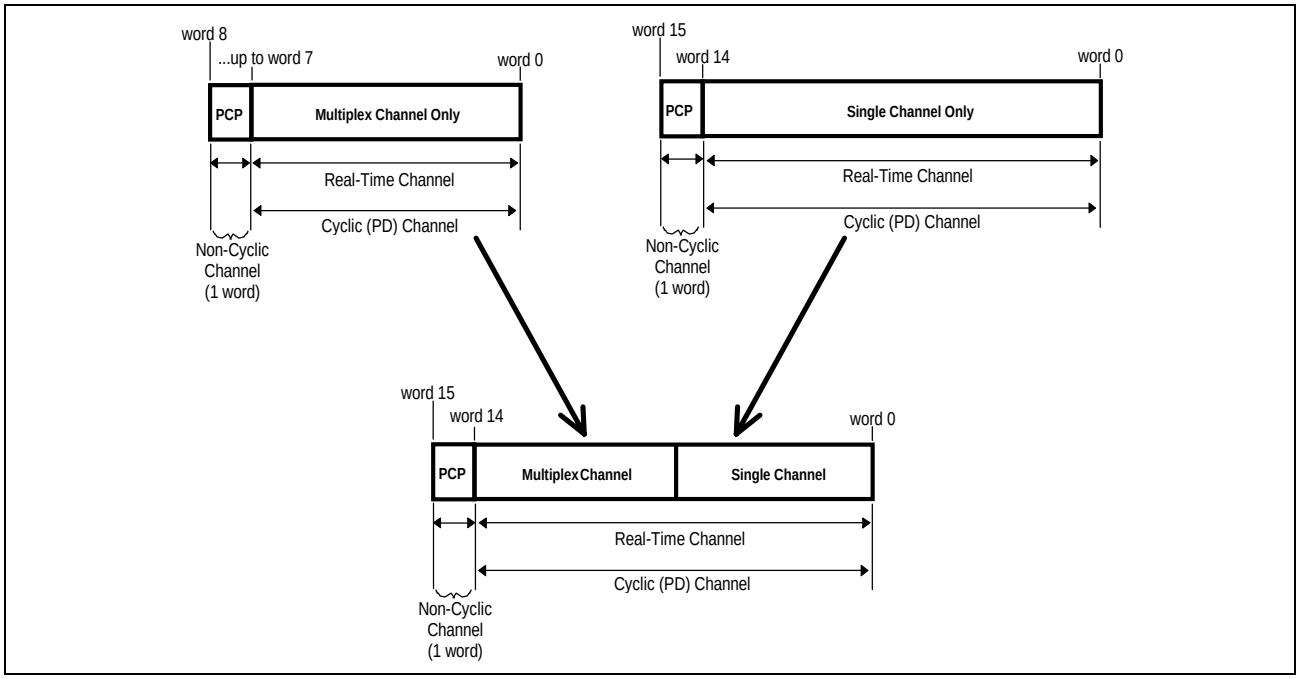


Figure 1-3: Configuration Options for the Cyclic Data Channel (3 possibilities)

The Real-Time Channel

In the real-time channel, data is updated cyclically between the Fieldbus master and slave. This channel contains two possible data types: **single** and **multiplex**.

Data Objects: Types and Sizes

The following table outlines the CLC-D data object types that can be transmitted via the cyclic channel and the amount of space (in 16-bit data words) that each data type consumes. Remember, the cyclic channel is limited to 16 data words in each direction (input and output).

Note: The cyclic data mapping list supports only 16- and 32-bit data of the following types for reading and writing:
- Integer
- Float
- Binary (used in CLC parameters)
- Hex (used in CLC parameters)

For all other data types (e.g. diagnostic messages - “strings”), use the non-cyclic Data Exchange Object or the Parameter Channel.

CLC Data Type	Data Size (in Data Words)
Register	1
Integer (currently active program ONLY *)	2
Float (currently active program ONLY *)	2
Global Integer	2
Global Float	2
CLC Card Parameter	2
CLC Axis Parameter	2
CLC Task Parameter	2
Note: Drive parameters "S" or "P" cannot be transmitted cyclically because of the inherent delay of parameter access over the SERCOS service channel. See ***"Non-Cyclic Channel/Data Exchange Objects"*** on page 1-8. However, if a drive parameter is mapped to an axis parameter, that axis parameter could be used in cyclic data (see description of Axis Parameters 180-196 in the **VisualMotion 6.0 Reference Manual**).	
*** Important Note:** Integers and floats are shown only for the currently active program. Each time you activate a new progam, the fieldbus reads/writes to the newly-activated program.	

Table 1-3: CLC-D Cyclic Data Types and Sizes

Single Data Types

Indramat Fieldbus interfaces have single (16-bit) and double (32-bit) word objects in the cyclic data channel with which simple applications can be handled without any difficulty. These objects are directly mapped to CLC-D data types and updated every third SERCOS update cycle.

Multiplex Data Types (Cyclic Data Channel)

In some multi-axis applications, 15 words of cyclic data transfer are not sufficient to meet the data transfer requirement of the application.

When insufficient data transfer space is available, multiplex data can be set up within the cyclic channel. One data object (base object) acts as a placeholder for multiple possible CLC-D data types (all of the same word size). The currently transmitted CLC-D data type is based on an index value placed in a multiplex control word attached to the end of the cyclic data list. Depending on the index specified by the master, the multiplex channel permits a different set of data within the cyclic channel to be transferred as current real-time data in both data directions.

Note: Using the multiplex channel will reduce the maximum number of usable words for storing CLC data to 14. The 15th word (or last used word, if fewer than 14 words) is used as the multiplex entry control/status word.

In the Fieldbus configuration lists, the following multiplex objects are available:

Type of Multiplex Object	Input Objects Available	Total Input Objects	Output Objects Available	Total Output Objects
32-bit	3 base (x16 indices)	48	3 base (x16 indices)	48
16-bit	2 base (x16 indices)	32	2 base (x16 indices)	32

Table 1-4: Available Multiplex Objects

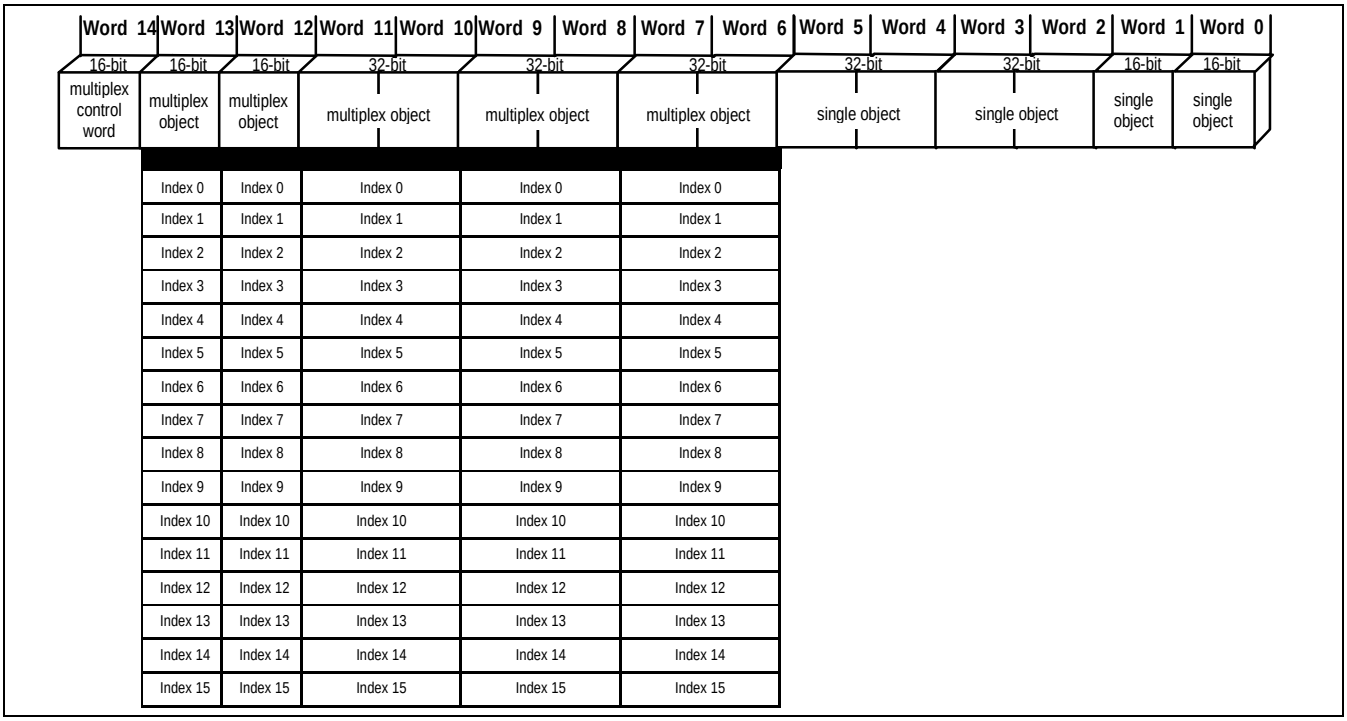

Word 14 Word 13 Word 12 Word 11 Word 10 Word 9 Word 8 Word 7 Word 6 Word 5 Word 4 Word 3 Word 2 Word 1 Word 0
16-bit 16-bit 16-bit 32-bit 32-bit 32-bit 32-bit 32-bit 16-bit 16-bit
multiplex control word
multiplex object
multiplex object
multiplex object
multiplex object
multiplex object
single object
single object
single object
single object
Index 0 Index 0 Index 0 Index 0 Index 0
Index 1 Index 1 Index 1 Index 1 Index 1
Index 2 Index 2 Index 2 Index 2 Index 2
Index 3 Index 3 Index 3 Index 3 Index 3
Index 4 Index 4 Index 4 Index 4 Index 4
Index 5 Index 5 Index 5 Index 5 Index 5
Index 6 Index 6 Index 6 Index 6 Index 6
Index 7 Index 7 Index 7 Index 7 Index 7
Index 8 Index 8 Index 8 Index 8 Index 8
Index 9 Index 9 Index 9 Index 9 Index 9
Index 10 Index 10 Index 10 Index 10 Index 10
Index 11 Index 11 Index 11 Index 11 Index 11
Index 12 Index 12 Index 12 Index 12 Index 12
Index 13 Index 13 Index 13 Index 13 Index 13
Index 14 Index 14 Index 14 Index 14 Index 14
Index 15 Index 15 Index 15 Index 15 Index 15

Figure 1-4: Sample Command (Write) to Slave (PLC→CLC)

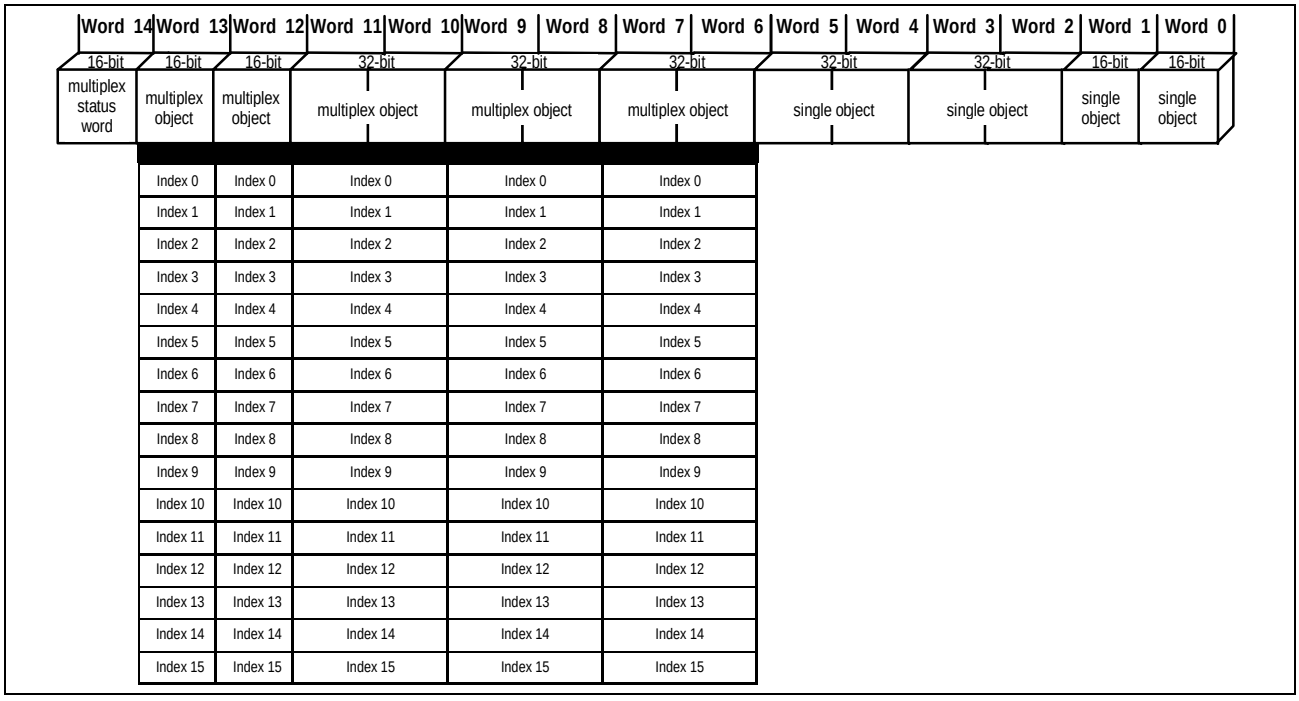

Word 14 Word 13 Word 12 Word 11 Word 10 Word 9 Word 8 Word 7 Word 6 Word 5 Word 4 Word 3 Word 2 Word 1 Word 0
16-bit 16-bit 16-bit 32-bit 32-bit 32-bit 32-bit 32-bit 16-bit 16-bit
multiplex status word
multiplex object
multiplex object
multiplex object
multiplex object
multiplex object
single object
single object
single object
single object
Index 0 Index 0 Index 0 Index 0 Index 0
Index 1 Index 1 Index 1 Index 1 Index 1
Index 2 Index 2 Index 2 Index 2 Index 2
Index 3 Index 3 Index 3 Index 3 Index 3
Index 4 Index 4 Index 4 Index 4 Index 4
Index 5 Index 5 Index 5 Index 5 Index 5
Index 6 Index 6 Index 6 Index 6 Index 6
Index 7 Index 7 Index 7 Index 7 Index 7
Index 8 Index 8 Index 8 Index 8 Index 8
Index 9 Index 9 Index 9 Index 9 Index 9
Index 10 Index 10 Index 10 Index 10 Index 10
Index 11 Index 11 Index 11 Index 11 Index 11
Index 12 Index 12 Index 12 Index 12 Index 12
Index 13 Index 13 Index 13 Index 13 Index 13
Index 14 Index 14 Index 14 Index 14 Index 14
Index 15 Index 15 Index 15 Index 15 Index 15

Figure 1-5: Sample Response (Read) to Master (CLC→PLC)

Multiplex Control and Status Words

The Interbus multiplex control and status words serve to command and acknowledge multiplex data transferred between the Fieldbus master and the Fieldbus slave. The **control** word is associated with **output**

communication (PLC→CLC). The **status** word is associated with **input** communication (CLC→PLC). Single data objects are not affected by the multiplex control and status words.

Note: For specific information about how the Fieldbus master uses the multiplex control and status words, see ***Multiplex Data Bits in the Control and Status Words*** on page 4-1.

Sending and Receiving the Same CLC-D Data Cyclically

Two separate objects will be consumed when sending and receiving the same CLC-D data (both cyclically and non-cyclically).

If you wish to cyclically send and receive the same CLC-D data, the Output mappings should come **first** in the list (see ***Fieldbus Mapper Examples, Basic Example*** on page 2-1).

Non-Cyclic Channel

The non-cyclic channel is used for data that needs to be transferred only once or sporadically, such as:

- the transmission of lists
- parametriziation of axes or programs
- any non-cyclically mapped data

Instead of being updated during each cycle, non-cyclic data is transferred in one-word increments over the PCP channel. The PCP channel is automatically inserted in the Interbus communications using the Indramat DBS boards. No initialization is required.

Though any data type can be transferred non-cyclically, diagnostic messages and drive parameters (S and P) **must** be transferred non-cyclically because of the non-cyclic retrieval for drive parameters through SERCOS and the length of the diagnostic messages.

For example: ASCII text in a diagnostic message requires one data word for every two ASCII characters. The non-cyclic channel can transfer only up to 16 data words (or 32 characters) of a diagnostic message at once. This would mean that one diagnostic text message (if 32 characters or more) could consume all the available cyclic data. For this reason, the transfer of diagnostic messages must be made over the non-cyclic (PCP) channel; it is not allowed over the real-time channel.

There are two types of non-cyclic data transmissions for the CLC-D/VisualMotion system:

- data transmitted via the data exchange object
- data mapped directly to CLC-D data types

Non-cyclic data can be accessed via:

- PCP support of the Fieldbus master

Direct-Mapped Data

Just as there are pre-defined cyclic data objects for mapping CLC data to the Interbus PD channel, there are also pre-defined non-cyclic (PCP) data objects (16- and 32-bit) that can be mapped to CLC data types. Non-cyclic objects have the following numeric names:

- **16-bit:** 5F60 - 5F7F and 5F90 - 5FBF
- **32-bit:** 5F00 - 5F1F

This type of data is mapped using the non-cyclic data mapping list (C-0-2700). It provides easy access to non-cyclic data. However, the fixed mapping list is limited to a certain number and types of CLC-D data.
The directly-mapped non-cyclic data (take note of size and direction) can be assigned to the following objects:

Number of Objects Available	Data Size	Direction (reference: Master Fieldbus)	Numeric Names of Objects
16	32-bit	in	5F10-5F1F
16	32-bit	out	5F00-5F0F
32	16-bit	in	5F60-5F6F, 5FA0-5FAF
32	16-bit	out	5F70-5F7F, 5Fb0-5FbF

Table 1-5: Non-Cyclic Data Objects

Note: The available CLC data types for directly-mapped non-cyclic data are the same as for directly-mapped cyclic data.

Data Object Types and Sizes

The following table outlines the directly-mapped CLC-D data object types that can be transmitted via the non-cyclic channel and the amount of space (in data words) that each data type consumes.

CLC Data Object Type	Data Size (in Data Words)
Register	1
Integer (currently active program ONLY)	2
Float (currently active program ONLY)	2
Global Integer	2
Global Float	2
CLC Card Parameter	2
CLC Axis Parameter	2
CLC Task Parameter	2

Table 1-6: Data Object Types/Sizes

Sending and Receiving the Same CLC-D Data Non-Cyclically

Two separate objects will be consumed when sending and receiving the same CLC-D data (both cyclically and non-cyclically).

Data Exchange Objects

Four data exchange objects 5E70 to 5E73 are available for the transfer of non-cyclic data. These objects represent fixed data "containers" of varying lengths that transfer the VisualMotion ASCII Protocol to the CLC-D card. These objects serve as an open-ended possibility to access any VisualMotion data (including cams, diagnostic text, etc.), but more work is required in the master to perform a transmission of this type. For more specific information about these objects, see ***Non-Cyclic Transmission (Data Exchange Objects)*** on page 4-6.

2 Fieldbus Mapper Examples

2.1 Basic Example

The following example demonstrates the process for configuring the mapping of basic data between the CLC-D card and an Interbus Fieldbus.

Important: The Fieldbus Mapper is an on-line configuration tool. The following conditions must be met when performing STEP I through STEP IV below:
- The system must be powered up.
- The CLC-D card must be connected to the system.
- The system must be set to parameter mode.

STEP I: Determine the Cyclic and Non-Cyclic Data

Cyclic Data

In this example, we will assume that you want to cyclically transfer the following data between the CLC-D card and the Fieldbus master in the order listed.

Note: Quantity and type of data varies depending on application.

Output Data (From Bus to CLC Card)	Size (Data Words)	Size (Bits)	Input Data (From CLC Card to Bus)	Size (Data Words)	Size (Bits)
Register 100	1	16	Integer 29	2	32
Float 2	2	32	Register 47	1	16
Global Integer 3	2	32	Register 120	1	16
Register 104	1	16	Global Float 22	2	32
Register 40	1	16	Axis Parameter 1.100	2	32
Integer 4	2	32	Integer 30	2	32
Integer 5	2	32			
TOTAL	11 Data Words			10 Data Words	

Table 2-1: Sample Cyclic Data

Non-Cyclic Data

We will assume that the following data will be transferred non-cyclically for system initialization or setup. Remember, the direct-mapped non-cyclic data is intended for infrequent data transfer.

Output Data (From Bus to CLC Card)	Size (Data Words)	Size (Bits)	Input Data (From CLC Card to Bus)	Size (Data Words)	Size (Bits)
Float 16	2	32	Float 5	2	32
Integer 7	2	32	Integer 8	2	32
Register 101	1	16			

Table 2-2: Sample Non-Cyclic Data

Note: See ***Table 1-5: Non-Cyclic Data Objects*** for data limitations.

Note: For other data types, such as diagnostic text messages, or S and P parameters, use the data exchange object (see ***Non-Cyclic Transmission (Data Exchange Objects)*** on page 4-6).

STEP II: Configure Fieldbus Slave Card with Setup Information and Cyclic Data Object Lists Using the Fieldbus Mapper

Important: The drive must be set to Parameter Mode to configure and assign the data mapping for Fieldbuses (STEP II and STEP III).

1. In the VisualMotion "Data" Menu, select "Fieldbus Mapper."

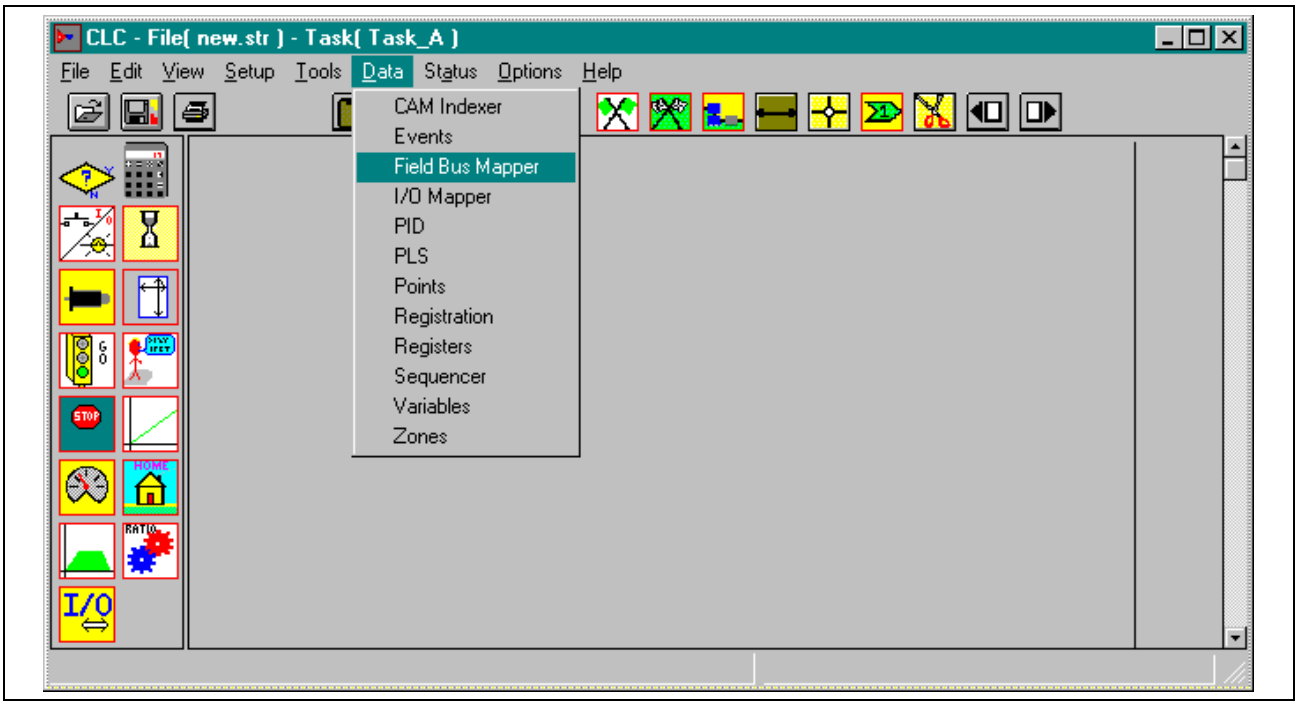


Figure 2-1: Selecting the Fieldbus Mapper from the main VisualMotion Screen

2. In the scroll box under "Bus Type," ensure that the desired bus is selected (If the DBS card is connected, Interbus should appear).
3. Choose the cyclic data channel (PD).

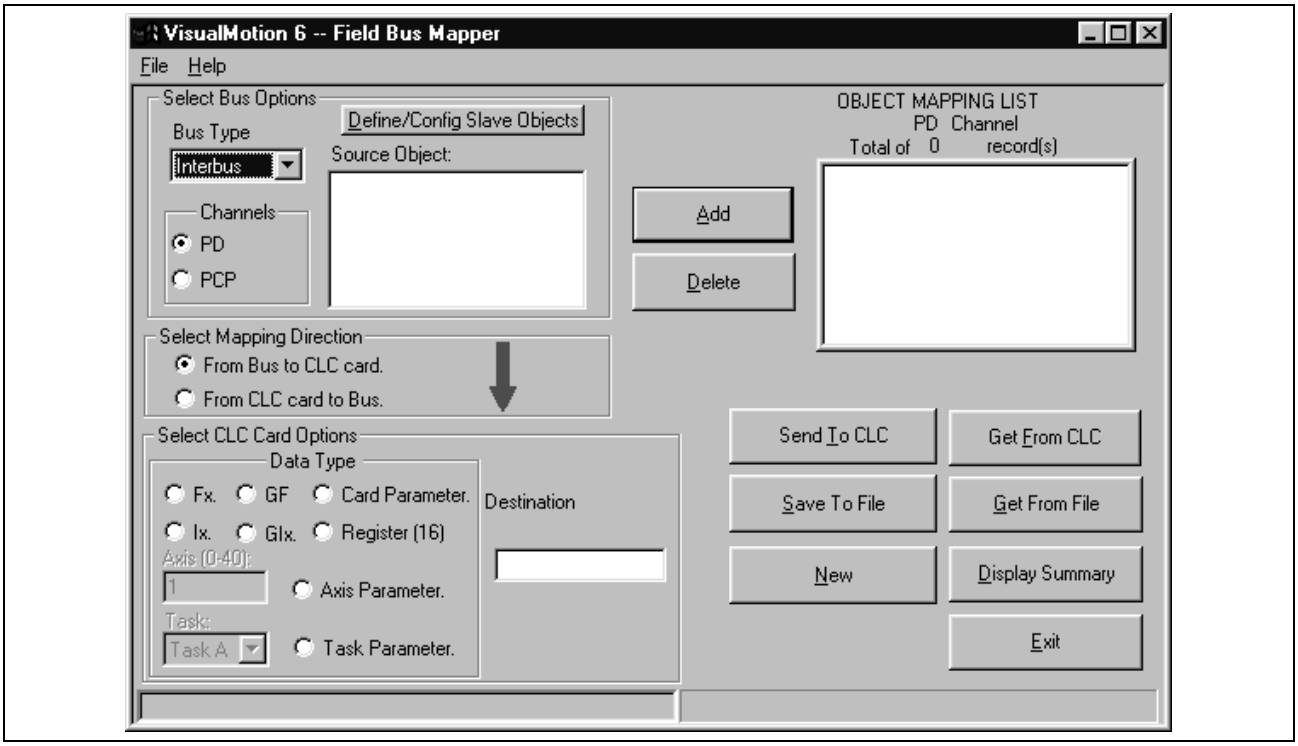


Figure 2-2: Main Fieldbus Mapper Screen

4. Click the button labeled "Define/Config Slave Objects." The Fieldbus Mapper will try to detect any configuration data from the CLC-D. If you are configuring a new list, click on the "NEW" button to clear any current data in the selected list. The Interbus Configuration Screen is pictured in ***Figure 2-3: Interbus Configuration Screen***.

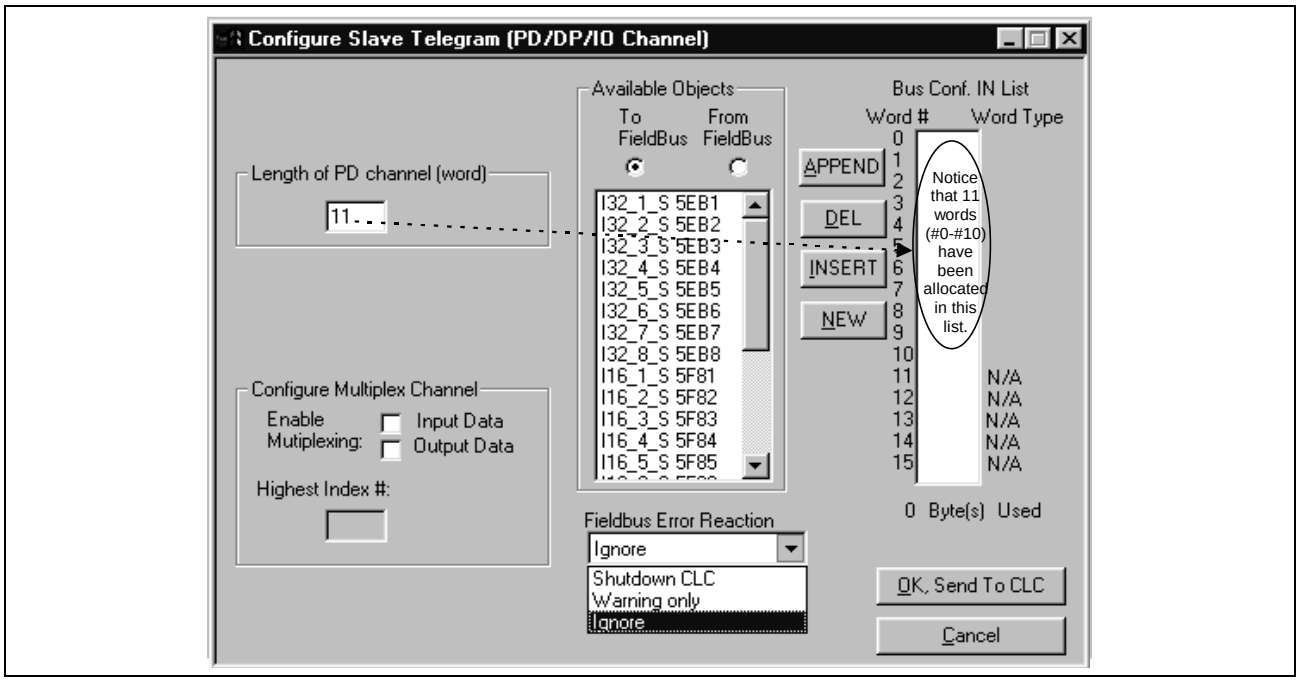


Figure 2-3: Interbus Configuration Screen

5. Set the "Length of PD Channel (Word)" to the number of data words to be allocated for the input channel or the output channel, whichever is greater.
   According to the example data in ***Table 2-1: Sample Cyclic Data*** under "STEP I: Determine the Cyclic and Non-Cyclic Data," we have set the length of the channel to 11 for the output channel (made up of [3] 16-bit objects of 1 data word each and [4] 32-bit objects of 2 data words each). The input channel contains only 10 words, so a “filler” object must be placed in the Bus Configuration IN list for the 11th word.
6. Ensure that both check boxes under "Configure Multiplex Channel" are unchecked. If you want to use multiplexing, see ***Multiplexing Example*** on page 2-10.
7. Ensure that the combo box under "Fieldbus Error Reaction" is currently set to "Ignore." Any other setting during setup will cause repeated error messages. Set the desired Error Reaction when the machine is ready for commissioning. See **Fieldbus Error Reaction** on page 3-9 for detailed information about the possible settings.
8. Choose the radio button below the words "To FieldBus."
9. For each data type desired, choose a corresponding available object in the list on the left and click "APPEND" to place it in the "Bus Conf. IN List." If you are configuring a new list, click on the "NEW" button to clear any current data in the selected list.
   Each available object has a typecode to identify its size and type. ***Figure 2-4: Configuring the Interbus Configuration IN List*** contains a description of the typecode.

Note: The desired order of the data on the master determines the object types (16- or 32-bit) and order of objects you place in the bus configuration list.

Note: Because the number of data objects in the IN list is less than the number in the OUT list, the remaining data word in the IN list must be assigned an available object that will not be used.

Following are descriptions of the buttons to manipulate the bus configuration lists:

inserts an available object after the last word placed in the current list.

removes the selected object from the current list.

inserts an available object above the selected object in the current list.

clears up the current list (only in the direction selected under “Available Objects”).

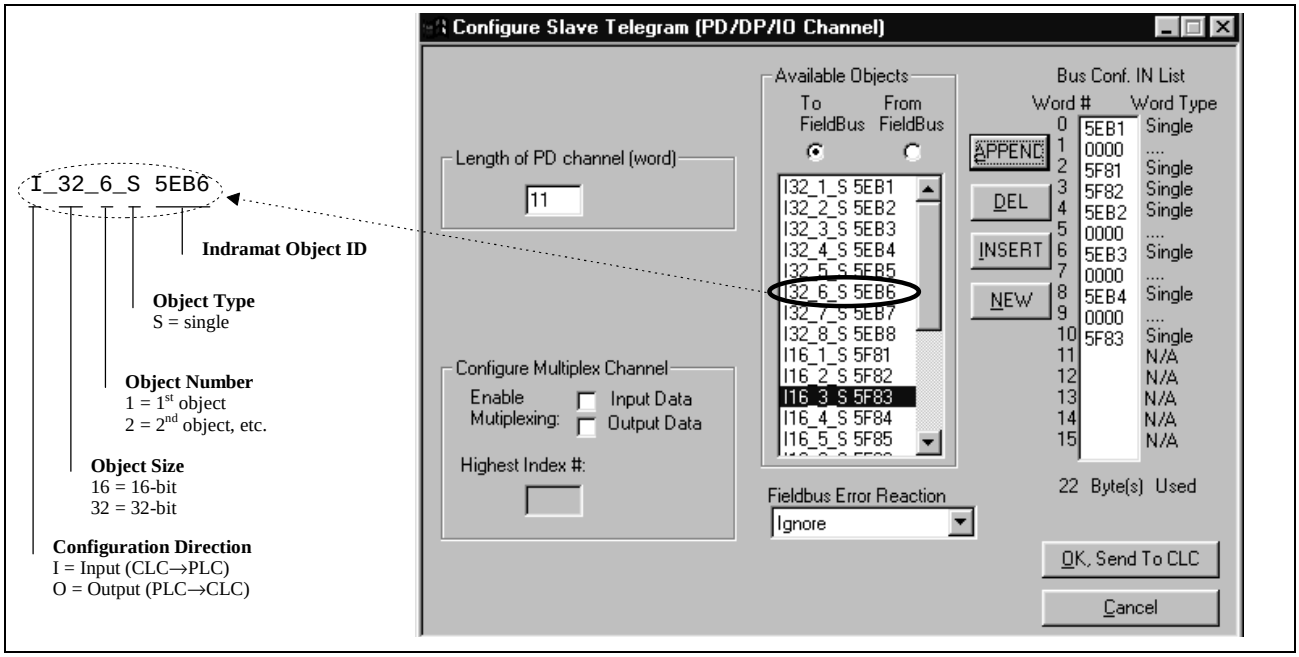


Figure 2-4: Configuring the Interbus Configuration IN List

12. Choose the radio button below the words "From FieldBus."

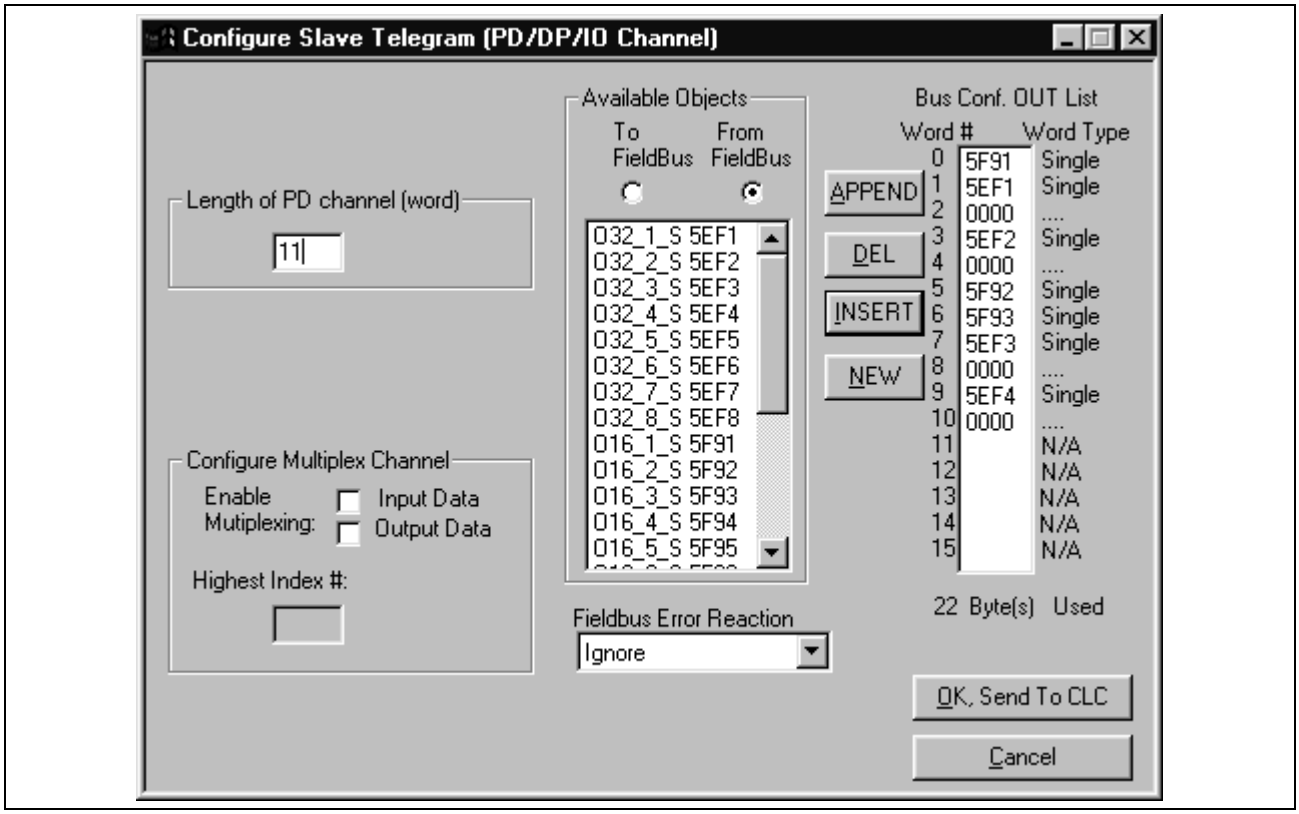


Figure 2-5: Configuring the Interbus Configuration OUT List

13. For each data type desired, choose a corresponding available object in the list on the left and click "APPEND" to place it in the "Bus Conf. OUT List." If you are configuring a new list, click on the "NEW" button to clear any current data in the selected list.

    We have now allocated the data objects in the cyclic channel so the desired data can be transferred between the CLC-D card and the Interbus Fieldbus card (see ***Figure 2-5: Configuring the Interbus Configuration OUT List***).

Note: The non-cyclic data requires no configuration of objects in a list. These objects are addressed directly and preassigned as non-cyclic. They need only to be mapped to the CLC-D data.

14. Click "OK, Send to CLC." The following window appears:

Figure 2-6: Warning after clicking "OK, Send to CLC"

STEP III: Define Cyclic Data Mapping Lists

STEP III associates the CLC-D data types to the objects assigned in STEP II.

In our example, the objects to be added are:

Output Data (From Bus to CLC Card)	Assigned Object	Input Data (From CLC Card to Bus)	Assigned Object
Register 100	5F91	Integer 29	5EB1
Float 2	5EF1	Register 47	5F81
Global Integer 3	5EF2	Register 120	5F82
Register 104	5F92	Global Float 22	5EB2
Register 40	5F93	Axis Parameter 1.100	5EB3
Integer 4	5EF3	Integer 30	5F83
Integer 5	5EF4		

Adding Objects to the Cyclic Data Mapping List

1. Ensure that the designated bus type is correct (Interbus).
2. Choose the desired cyclic data channel (PD).
3. Select the desired Mapping Direction.
   Our example begins with "From Bus to CLC Card."
4. Select the Data Type.
   In our example, the first item to be added to this list is Register 100.
   Select "Register" as the data type.

Note: The CLC-D will show only the objects associated with the chosen data type in the Source Object list. For example, if you have selected the mapping direction "From Bus to CLC Card" and designate "Register" as the data type, only 16-bit output objects that have been mapped in the bus configuration list (see STEP II) will be shown in the Source Object list.

5. Select or fill in the Destination. In our example, we must select Register 100 from the list in the pop-up screen shown in ***Figure 2-7: Sample Selection List*** OR type 100 as the destination.

Note: The Fieldbus Mapper selection lists display only the labels of the currently active program.

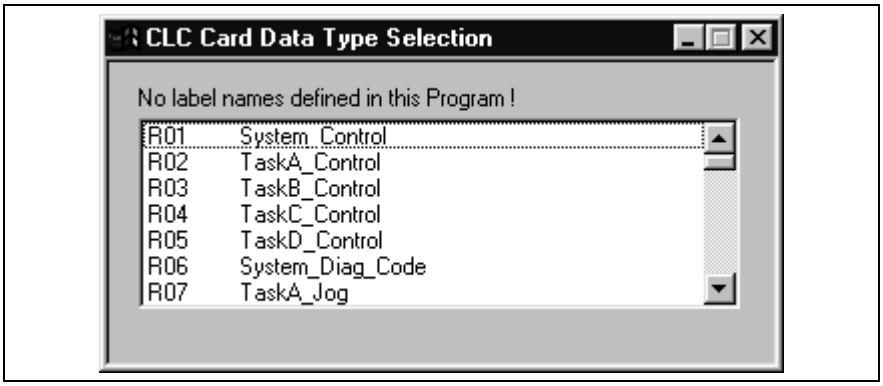


Figure 2-7: Sample Selection List

6. The source objects that are applicable to this data type will appear in the Source Object list. Choose the first available object.
7. Click the "Add" button. The object will appear in the "OBJECT MAPPING LIST" at the right.

Note: This step associates the object assigned in STEP II to the CLC-D data. The CLC-D scans the list created in this step and executes each mapping instruction **in order** during every third SERCOS update cycle.

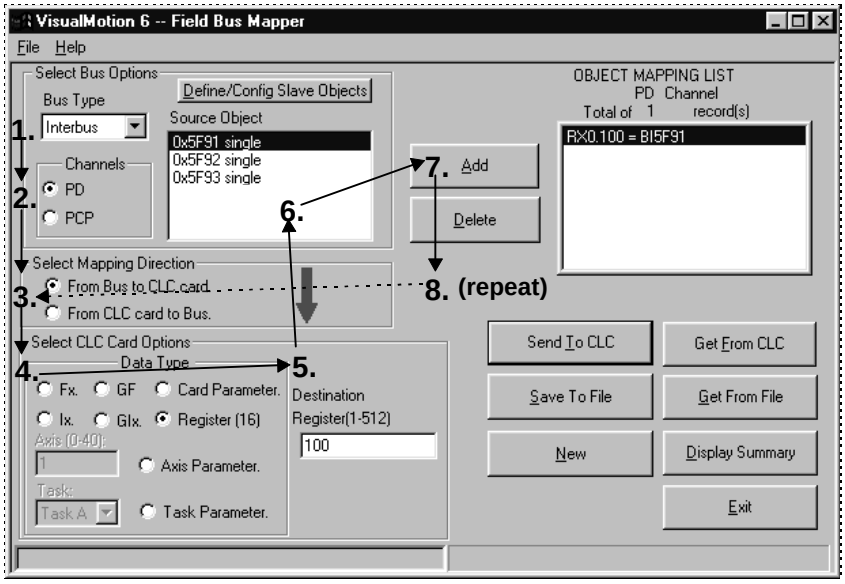


Figure 2-8: Adding Objects to the Cyclic Object Mapping List

Important: Cyclic data uses the objects configured in STEP II.

8. Repeat steps 3 through 7 above until each of the desired cyclic data objects has been added to the OBJECT MAPPING LIST.

Note: When mapping a PLC output object to a parameter, the parameter must be of the type "Read/Write Any Time." For parameters that are "Read Any Time/Write in Parameter Mode" or "Read Only," an error message is generated when you click “Send to CLC” if you try to map the parameters to a PLC output object. Examples: C-0-1550 (Read Any Time / Write in Parameter Mode) and C-0-1518 (Read Only).

9. Click “Send to CLC” to save the selected object mapping list to the CLC-D card. Use any of the following buttons, as needed.

Send To CLC	Saves the currently selected object mapping list to the CLC-D Card (to card parameter C-0-2600 if the cyclic [PD] channel is selected, and C-0-2700 if the non-cyclic [PCP] channel is selected).
Save To File	Saves the currently selected object mapping list to a file (with a .prm extension).
New	Clears up the current object mapping list.
Get From CLC	Gets the currently selected object mapping list from the CLC-D Card (from card parameter C-0-2600 if the cyclic [PD] channel is selected, and C-0-2700 if the non-cyclic [PCP] channel is selected).
Get From File	Allows the user to open a previously saved file (with a .prm extension) of an object mapping list. When the file is opened, displays that object mapping list.
Display Summary	Generates an on-line summary report of the currently selected Fieldbus. This report can be printed by selecting "Print" from the File menu in the Fieldbus Mapper window.
Exit	Exits the Fieldbus Mapper utility.

Important: Ensure that your system is set to Parameter Mode for the changes to take place.

Note: For debugging purposes, the Fieldbus Mapper follows each mapped item with the cursor as it is saved to the CLC-D. If there is a problem with the mapping of a particular object, an error message appears while the cursor remains on the object.

Changing an Existing Object Mapping List

If you want to insert, replace or delete items in the Object Mapping List, double-click on the item in question. A new set of buttons appears to the left of the Object Mapping List box.

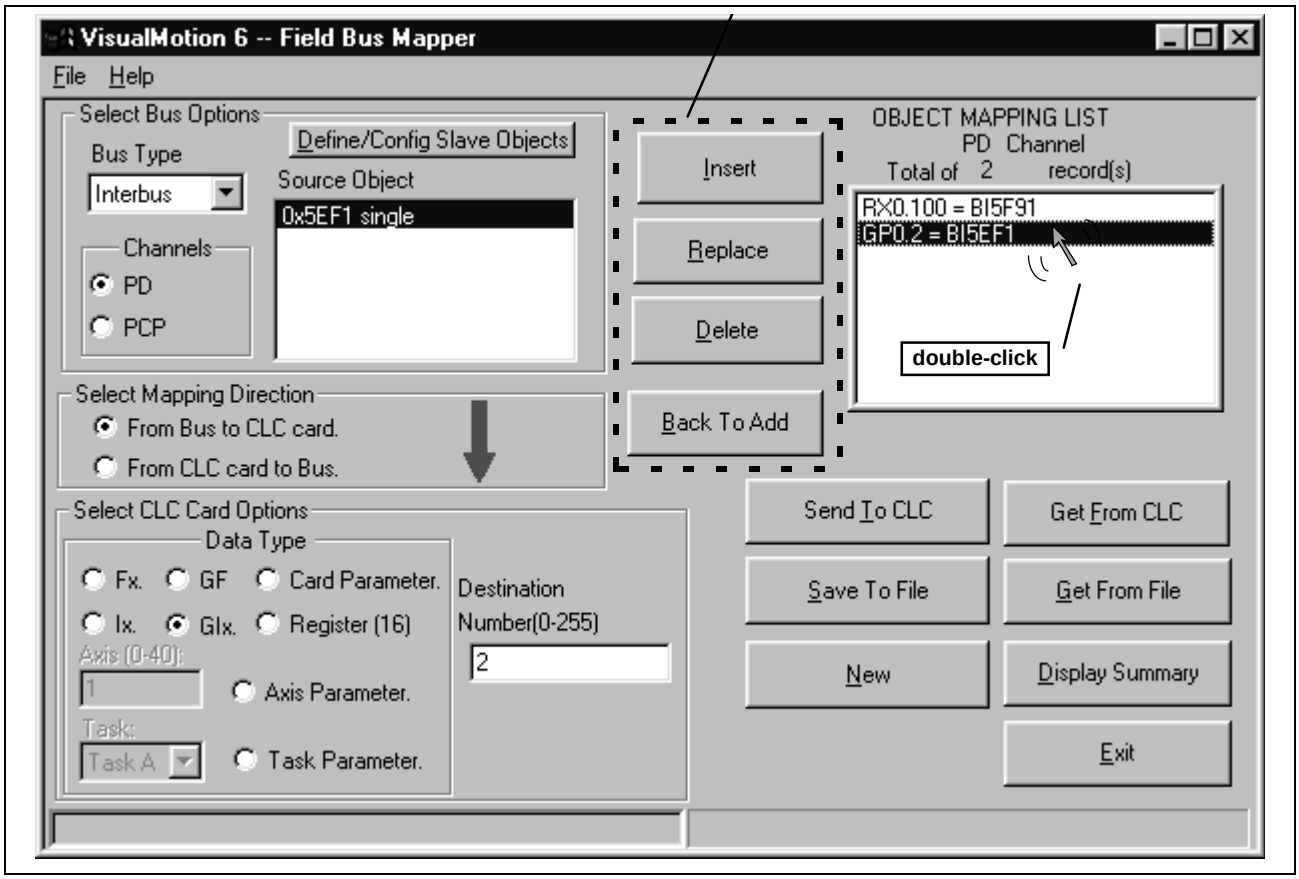


Figure 2-9: Changing an Existing Mapping List in the Fieldbus Mapper

Insert	Inserts a new object into the list directly before the selected object.
Replace	Replaces the selected object with a new object.
Delete	Removes the selected object from the list.
Back To Add	Returns to Fieldbus mapper normal mode, which allows adding new items to the end of the list and deleting items.

STEP IV: Define Non-Cyclic Data Mapping Lists (Direct Mapping)

The non-cyclic data uses objects that are pre-defined for this type of data.

In our example, the following data will be transmitted non-cyclically:

Output Data (From Bus to CLC Card)	Input Data (From CLC Card to Bus)
Float 16	Float 5
Integer 7	Integer 8
Register 101	

Table 2-3: Objects to be transferred non-cyclically

Adding Objects to the Non-Cyclic Data Mapping List (see *Figure 2-10: Adding Objects to the Non-Cyclic Object Mapping List*)

1. Ensure that the designated bus type is correct.
2. Choose the non-cyclic data channel (PCP).
3. Select the desired Mapping Direction.
   In our example, we will begin with the output list, or "From Bus to CLC."
4. Select the Data Type.
   In our example, the first item to be added to this list is Float 16. Select "Float" as the data type.
5. Fill in the Destination.
   In our example, we must type 16 as the destination.
6. The source objects that are applicable to this data type will appear in the Source Object list.
   Choose the first available object.
7. Click the "Add" button.
   The object will appear in the "OBJECT MAPPING LIST" at the right. This list is scanned by the CLC only when a non-cyclic request comes in and executes the mapping.
8. Repeat steps 2 through 7 above until each of the desired non-cyclic objects has been added to the OBJECT MAPPING LIST in both directions ("From Bus to CLC card" AND "From CLC card to Bus").

Note: If you want to insert, replace or delete items in the Object Mapping List, double-click on the item in question. A new set of buttons appears to the left of the Object Mapping List box. See ***Changing an Existing Object Mapping List*** on page 2-7 for a detailed explanation of each button.

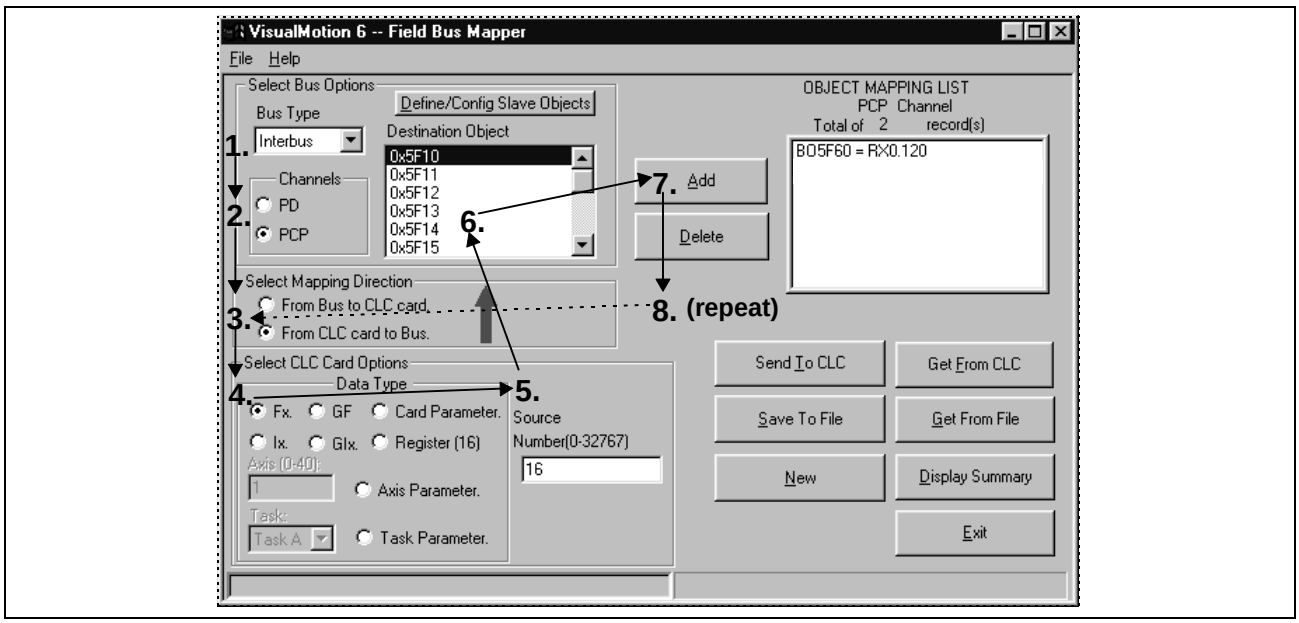


Figure 2-10: Adding Objects to the Non-Cyclic Object Mapping List

9. Click “Send to CLC” to save the selected object mapping list to the CLC-D card. Use any of the following buttons, as needed.

Saves the currently selected object mapping list to the CLC-D Card (to card parameter C-0-2600 if the cyclic [PD] channel is selected, and C-0-2700 if the non-cyclic [PCP] channel is selected).

Saves the currently selected object mapping list to a file (with a .prm extension).

Clears up the current object mapping list.

Gets the currently selected object mapping list from the CLC-D Card (from card parameter C-0-2600 if the cyclic [PD] channel is selected, and C-0-2700 if the non-cyclic [PCP] channel is selected).

Allows the user to open a previously saved file (with a .prm extension) of an object mapping list. When the file is opened, displays that object mapping list.

Generates an on-line summary report of the currently selected Fieldbus. This report can be printed by selecting "Print" from the File menu in the Fieldbus Mapper window.

Exits the Fieldbus Mapper utility.

Note: For debugging purposes, the Fieldbus Mapper follows each mapped item with the cursor as it is saved to the CLC-D. If there is a problem with the mapping of a particular object, an error message appears while the cursor remains on the object.

10. **Mapping is complete!**
For programming information, refer to ***Information for the GPS Programmer*** on page 3-1or ***Information for the PLC Programmer*** on page 4-1.

2.2 Multiplexing Example

Multiplexing is available in the cyclic (PD) channel for applications where more than the allotted 15 data transfer words are required. For a more detailed description of multiplexing, see ***Multiplex Data Types*** (Cyclic Data Channel) on page 1-4.

Important: The Fieldbus Mapper is an on-line configuration tool. The following conditions must be met when performing STEP I through STEP IV below:
- The system must be powered up.
- The CLC-D card must be connected to the system.
- The system must be set to parameter mode.

STEP I: Determining the Cyclic and Non-Cyclic Data (with Multiplexing)

Cyclic Data

A typical multiplexing example in a multi-axis application is to assign cyclic data to each index by axis (e.g. index 0 to axis 1, index 1 to axis 2, etc.). Although this example shows data for only three axes in this manner, remember that multiplexing allows sending up to 16 unique pieces of data for each multiplex object. Single and multiplex objects can be combined to fill up the first 14 words of the list.

In this example, we will assume that you want to cyclically transfer the following data between the CLC-D card and the Fieldbus master in the order listed:

Object Type, Size (bits)	Output Data (From Fieldbus Master to CLC Card)			Size (Data Words)	Object Type, Size (bits)	Input Data (From CLC Card to Fieldbus Master)			Size (Data Words)
	Axis 1 (Index 0)	Axis 2 (Index 1)	Axis 3 (Index 2)			Axis 1 (Index 0)	Axis 2 (Index 1)	Axis 3 (Index 2)	
Single, 16-bit	Register 100			1	Single, 16-bit	Register 120			1
Single, 32-bit	Float 2			2	Single, 32-bit	Global Float 22			2
Single, 32-bit	Integer 4			2	Single, 32-bit	Global Integer 44			2
Multiplex, 16-bit	Control Registers (preassigned per axis)				Multiplex, 16-bit	Status Registers (preassigned per axis)			
	Register 11	Register 12	Register 13	1		Register 31	Register 32	Register 33	1
Multiplex, 32-bit	Position Command (Floats called out in VisualMotion program)*see Note				Multiplex, 32-bit	Axis Parameter 102 (Feedback Position)			
	Float 11	Float 21	Float 31	2		Parameter A-1.102	Parameter A-2.102	Parameter A-3.102	2
Multiplex, 32-bit	Counter (Integers called out in VisualMotion program)*see Note				Multiplex, 32-bit	Axis Parameter 112 (Feedback Velocity)			
	Integer 11	Integer 21	Integer 31	2		Parameter A-1.100	Parameter A-2.100	Parameter A-3.100	2
Multiplex, 32-bit	Dwell Time (Integers called out in VisualMotion program)*see Note				Multiplex, 32-bit	Maximum Attempts (Integers called out in VisualMotion program) *see Note			
	Integer 14	Integer 24	Integer 34	2		Integer 13	Integer 23	Integer 33	2
				TOTAL: 12 Data Words					**TOTAL:** 12 Data Words

Note: We suggest devising a system for assigning multiplex integers and floats to particular axes or data sets. (However, there is no limitation on how to divide multiplex objects, except size.)In our sample data, the system is as follows:
1st digit: Axis number
2nd digit: Purpose (e.g. position, counter, dwell)

For example: Integers in above sample data

3 4

1=Axis 1, 2=Axis 2, 3=Axis 3... 1=Counter, 4=Dwell Time

Table 2-4: Sample Cyclic Data (with Multiplexing)

Non-Cyclic Data

We will assume that the following data will be transferred non-cyclically for system initialization or setup. Remember, the direct-mapped non-cyclic data is intended for infrequent data transfer.

Note: See ***Table 1-5: Non-Cyclic Data Objects*** for data limitations.

Output Data (From Fieldbus Master to CLC Card)	Size (Data Words)	Size (Bits)	Input Data (From CLC Card to Fieldbus Master)	Size (Data Words)	Size (Bits)
Float 16	2	32	Float 5	2	32
Integer 7	2	32	Integer 8	2	32
Register 101	1	16			

Table 2-5: Sample Non-Cyclic Data (with Multiplexing)

Note: For other data types, such as diagnostic text messages, or S and P parameters, use the data exchange object.

STEP II: Configure Fieldbus Slave Card with Setup Information and Cyclic Data Object Lists Using the Fieldbus Mapper

Important: The drive must be set to Parameter Mode to configure and assign the data mapping for Fieldbuses (STEP II and STEP III).

1. In the VisualMotion "Data" Menu, select "Fieldbus Mapper."

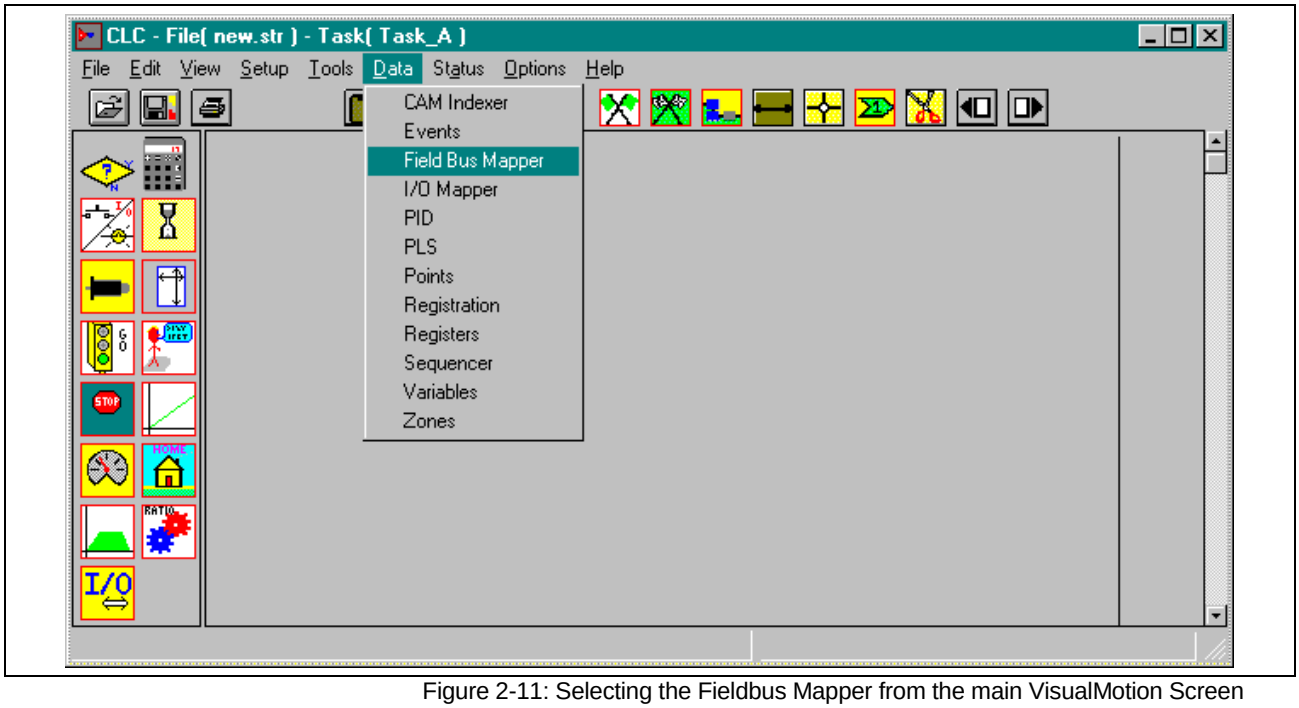


Figure 2-11: Selecting the Fieldbus Mapper from the main VisualMotion Screen

2. In the scroll box under "Bus Type," choose the desired bus (Interbus). The bus type may appear automatically if the Fieldbus card is connected and the firmware version is recognized.
3. Choose the cyclic data channel (PD).

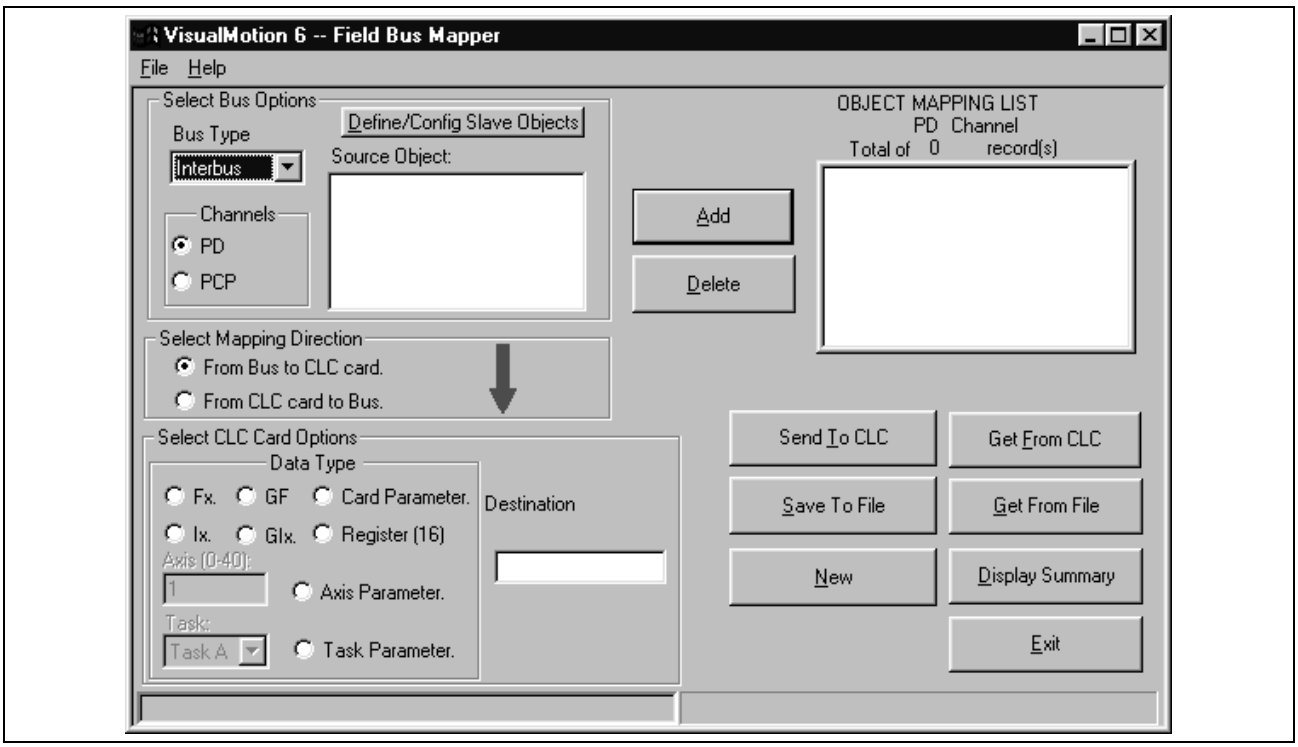


Figure 2-12: Main Fieldbus Mapper Screen

4. Click the button labeled "Define/Config Slave Objects." The Fieldbus Mapper will try to detect any configuration data from the CLC-D. If you are configuring a new list, click on the "NEW" button to clear any current data in the selected list. The Interbus Configuration Screen is pictured in ***Figure 2-13: Interbus Configuration Screen***.

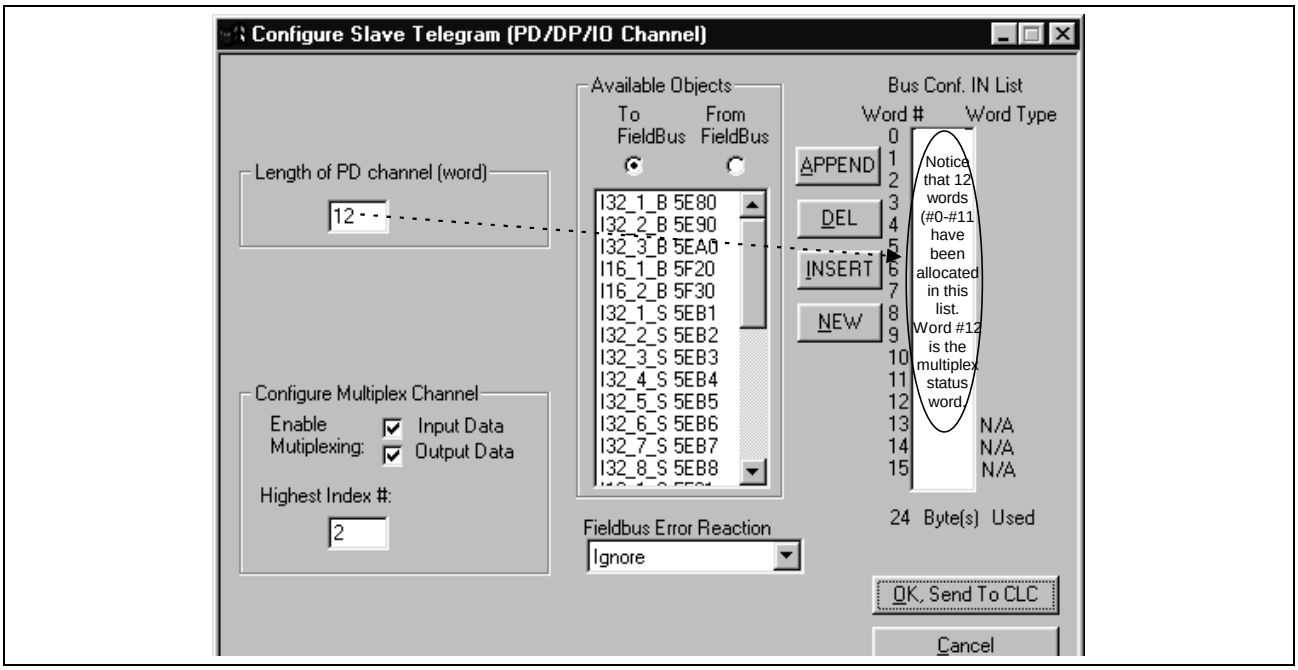


Figure 2-13: Interbus Configuration Screen

5. Set the "Length of PD Channel (Word)" to the number of data words to be allocated for the input channel or the output channel, whichever is greater. According to the example data in ***Table 2-4: Sample Cyclic Data (with Multiplexing)*** under "STEP I: Determine the Cyclic and Non-Cyclic Data," we have set the length of the channel to 12 for the output and input channels.
6. Click in both check boxes next to "Enable Multiplexing," because we want to configure multiplex objects in both the output and input channels.

Notice that the 13th word (word #12) shows the status/control word (STAT in the Bus Conf. IN list, CNTL in the Bus Conf. OUT list). The bus now actually has 13 words in each of the bus configuration lists: the 12 words that were set on the left, plus the status/control word. See ***Multiplex Control and Status Words*** on page 1-5 for information about the usage of this extra word when accessing data from the Fieldbus master (PLC).

7. Fill in the highest index number as "2," because we are using only 3 sets of multiplex data (Indexes 0-2, for axes 1-3). The index number will always be one less than the number of indices, because the first index number is "0."
8. Ensure that the combo box under "Fieldbus Error Reaction" is currently set to "Ignore." Any other setting during setup will cause repeated error messages. Set the desired Error Reaction when the machine is ready for commissioning. See **Fieldbus Error Reaction** on page 3-9 for detailed information about the possible settings.
9. Choose the radio button below the words "To FieldBus."
10. Click “NEW” to clear up any existing data in the current list.

Note: The desired order of the data on the master determines the object types (16- or 32-bit) and order of objects you place in the bus configuration list. Therefore, care should be taken when placing these objects.

---

11. For each data type desired, choose a corresponding available object in the list on the left and click "APPEND" to place it at the end of the "Bus Conf. IN List."
    Each available object has a typecode to identify its size and type.
    Figure 2-14: Configuring the Interbus Configuration IN List contains a description of the typecode.

Following are descriptions of the buttons to manipulate the bus configuration lists:

APPEND	inserts an available object after the last word placed in the current list.
DEL	removes the selected object from the current list.
INSERT	inserts an available object above the selected object in the current list.
NEW	clears up the current list (only in the direction selected under “Available Objects”).

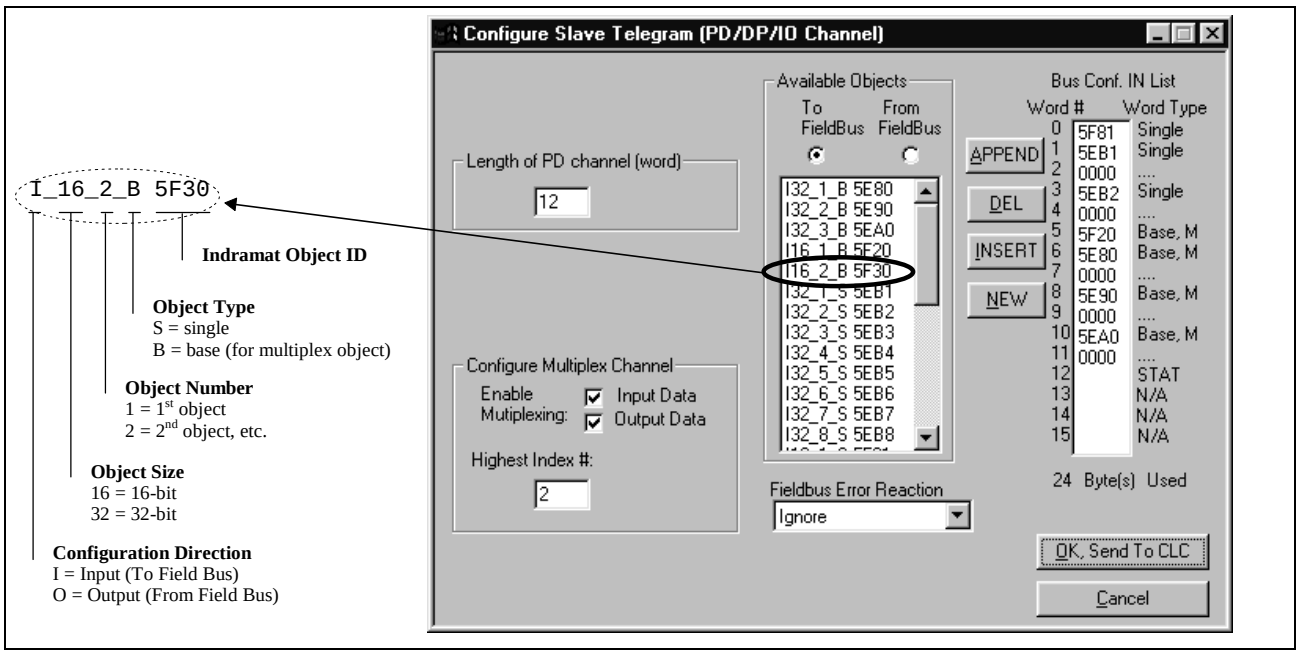


Figure 2-14: Configuring the Interbus Configuration IN List (with Multiplexing)

12. Choose the radio button below the words "From FieldBus."

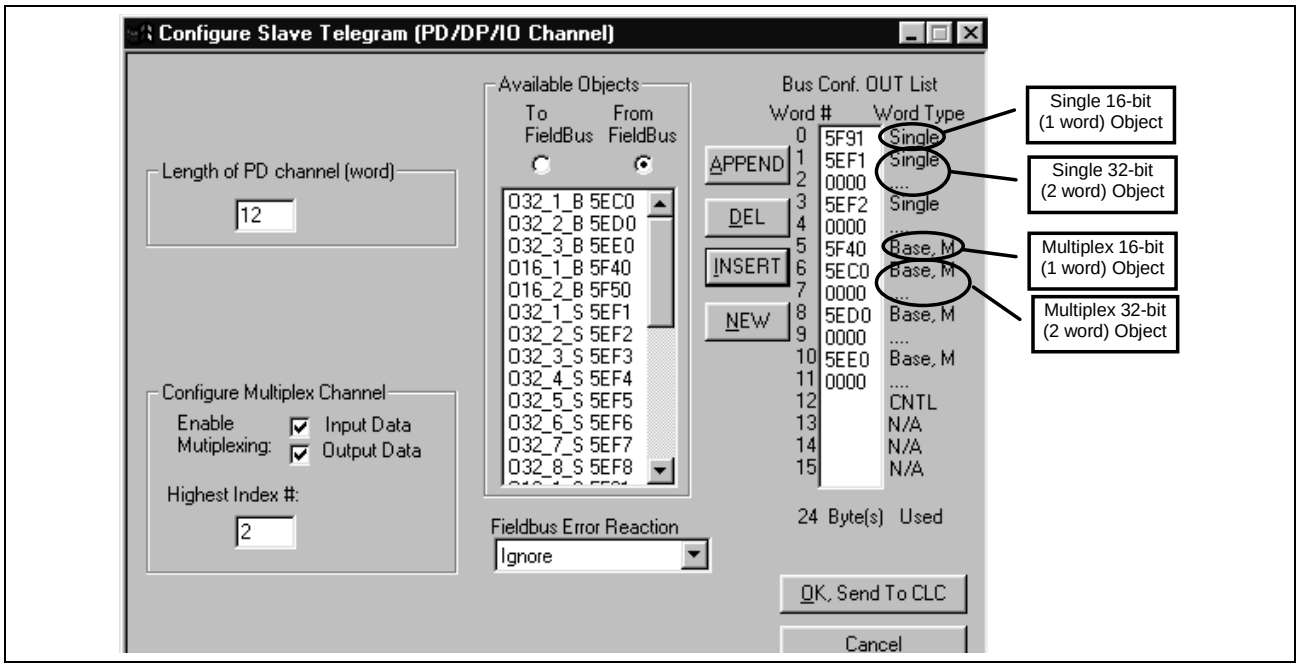


Figure 2-15: Configuring the Interbus Configuration OUT List (with Multiplexing)

13. For each data type desired, choose a corresponding available object in the list on the left and click "APPEND" to place it in the "Bus Conf. OUT List."

We have now allocated the data objects in the cyclic channel so the desired data can be transferred between the CLC-D card and the Interbus Fieldbus card.

Note: The non-cyclic data requires no configuration of objects in a list. These objects are addressed directly and preassigned as non-cyclic. They need only to be mapped to the CLC-D data.

---

14. Click "OK, Send to CLC." The following window appears:

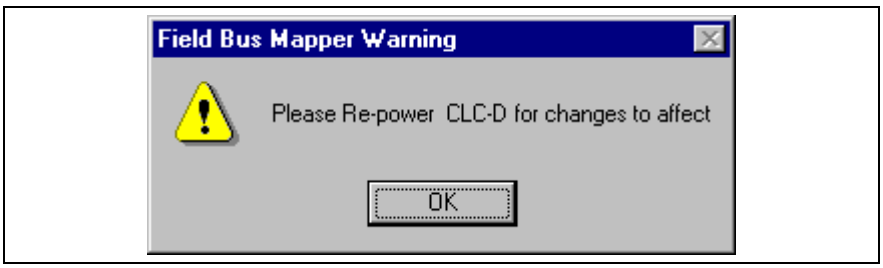


Figure 2-16: Warning after clicking "OK, Send to CLC"

STEP III: Define Cyclic Data Mapping Lists

STEP III associates the CLC-D data types to the objects assigned in STEP II. In our example, the objects to be added are:

Object Type	Output Data (From Bus to CLC Card)			Input Data (From CLC Card to Bus)		
Single	Register 100			Register 120		
"	Float 2			Global Float 22		
"	Integer 4			Global Integer 44		
Multiplex	Register 11	Register 12	Register 13	Register 31	Register 32	Register 33
"	Float 11	Float 21	Float 31	Parameter A-1.102	Parameter A-2.102	Parameter A-3.102
"	Integer 11	Integer 21	Integer 31	Parameter A-1.110	Parameter A-2.110	Parameter A-3.110
"	Integer 14	Integer 24	Integer 34	Integer 13	Integer 23	Integer 33

Table 2-6: Objects to be transferred cyclically (with multiplexing)

Adding Objects to the Cyclic Data Mapping List

1. Ensure that the designated bus type is correct.
2. Choose the desired cyclic data channel (PD).
3. Select the desired Mapping Direction.
   Our example begins with "From Bus to CLC Card."
4. Select the Data Type.
   In our example, the first single item to be added to this list is Register 120. Select "Register" as the data type.

Note: The CLC-D will show only the objects associated with the chosen data type in the Source Object list. For example, if you have selected the mapping direction "From Bus to CLC Card" and designate "Register" as the data type, only 16-bit output objects that have been mapped in the bus configuration list (see STEP II) will be shown in the Source Object list.

5. Select or fill in the Destination.
   In our example, we must select Register 120 from the list in the pop-up screen shown in ***Figure 2-7: Sample Selection List***, OR type 120 as the destination.

Note: The Fieldbus Mapper selection lists display only the labels of the currently active program.

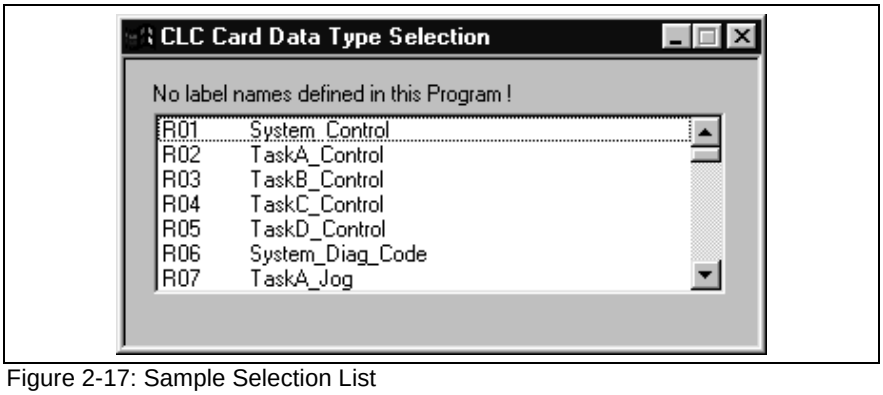


Figure 2-17: Sample Selection List

6. The source objects that are applicable to this data type will appear in the Source Object list.
   Choose the first available object.
7. Click the "Add" button.
   The object will appear in the "OBJECT MAPPING LIST" at the right.

Note: This step associates the object assigned in STEP II to the CLC-D data. The CLC-D scans the list created in this step and executes each mapping instruction **in order** every third SERCOS update cycle. Multiplex data can only be updated when this index is commanded by the multiplex control word.

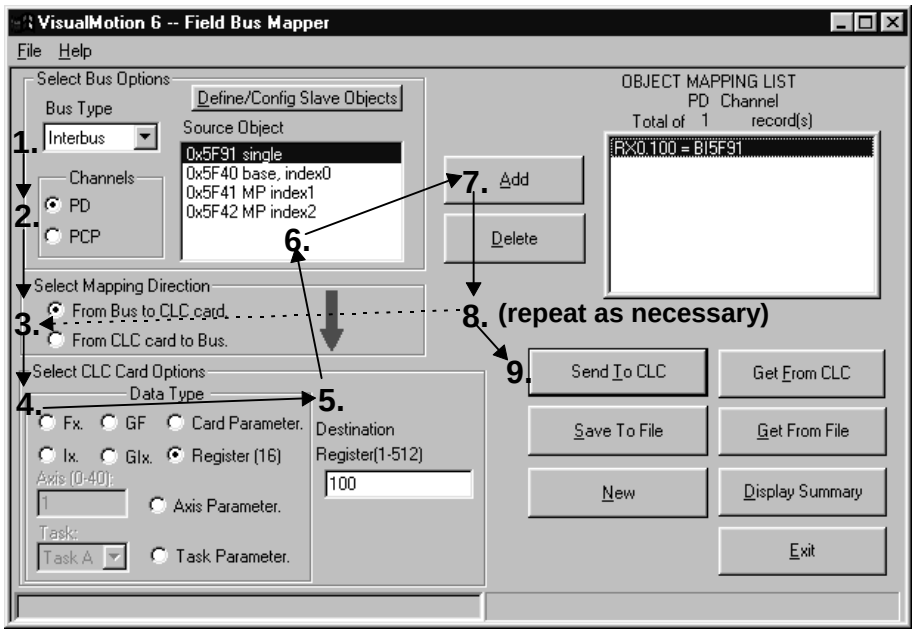


Figure 2-18: Adding Objects to the Cyclic Object Mapping List

Important: Cyclic data uses the objects configured in STEP II.

8. Repeat steps 3 through 7 above until each of the desired cyclic data objects has been added to the OBJECT MAPPING LIST.

Note: When mapping a PLC output object to a parameter, the parameter must be of the type "Read/Write Any Time." For parameters that are "Read Any Time/Write in Parameter Mode" or "Read Only," an error message is generated when you click “Send to CLC” if you try to map the parameters to a PLC output object. Examples: C-0-1550 (Read Any Time / Write in Parameter Mode) and C-0-1518 (Read Only).

Note: If you want to insert, replace or delete items in the Object Mapping List, double-click on the item in question. A new set of buttons appears to the left of the Object Mapping List box. See ***Changing an Existing Object Mapping List*** on page 2-7 for a detailed explanation of each button.

9. Click “Send to CLC” to save the selected object mapping list to the CLC-D card. Use any of the following buttons, as needed.

Saves the currently selected object mapping list to the CLC-D Card (to card parameter C-0-2600 if the cyclic [PD] channel is selected, and C-0-2700 if the non-cyclic [PCP] channel is selected).

Saves the currently selected object mapping list to a file (with a .prm extension).

Clears up the current object mapping list.

Gets the currently selected object mapping list from the CLC-D Card (from card parameter C-0-2600 if the cyclic [PD] channel is selected, and C-0-2700 if the non-cyclic [PCP] channel is selected).

Allows the user to open a previously saved file (with a .prm extension) of an object mapping list. When the file is opened, displays that object mapping list.

Generates an on-line summary report of the currently selected Fieldbus. This report can be printed by selecting "Print" from the File menu in the Fieldbus Mapper window.

Exits the Fieldbus Mapper utility.

Important: Ensure that your system is set to Parameter Mode for the changes to take place.

STEP IV: Define Non-Cyclic Data Mapping Lists (Direct Mapping)

The non-cyclic data uses objects that are pre-defined for this type of data.

In our example, the following data will be transmitted non-cyclically:

Output Data (From Bus to CLC Card)	**Input Data (From CLC Card to Bus)**
Float 16	Float 5
Integer 7	Integer 8
Register 101	

Table 2-7: Objects to be transferred non-cyclically

Adding Objects to the Non-Cyclic Data Mapping List (see *Figure 2-19: Adding Objects to the Non-Cyclic Object Mapping List*)

1. Ensure that the designated bus type is correct.
2. Choose the non-cyclic data channel (PCP).
3. Select the desired Mapping Direction.
   In our example, we will begin with the output list, or "From Bus to CLC."
4. Select the Data Type.
   In our example, the first item to be added to this list is Float 16. Select "Float" as the data type.
5. Fill in the Destination.
   In our example, we must type 16 as the destination.
6. The source objects that are applicable to this data type will appear in the Source Object list. Choose the first available object.
7. Click the "Add" button.
   The object will appear in the "OBJECT MAPPING LIST" at the right. This list is scanned by the CLC only when a non-cyclic request comes in and executes the mapping.
8. Repeat steps 2 through 7 above until each of the desired non-cyclic objects has been added to the OBJECT MAPPING LIST in both directions ("From Bus to CLC card" AND "From CLC card to Bus").

Note: If you want to insert, replace or delete items in the Object Mapping List, double-click on the item in question. A new set of buttons appears to the left of the Object Mapping List box. See ***Changing an Existing Object Mapping List*** on page 2-7 for a detailed explanation of each button.

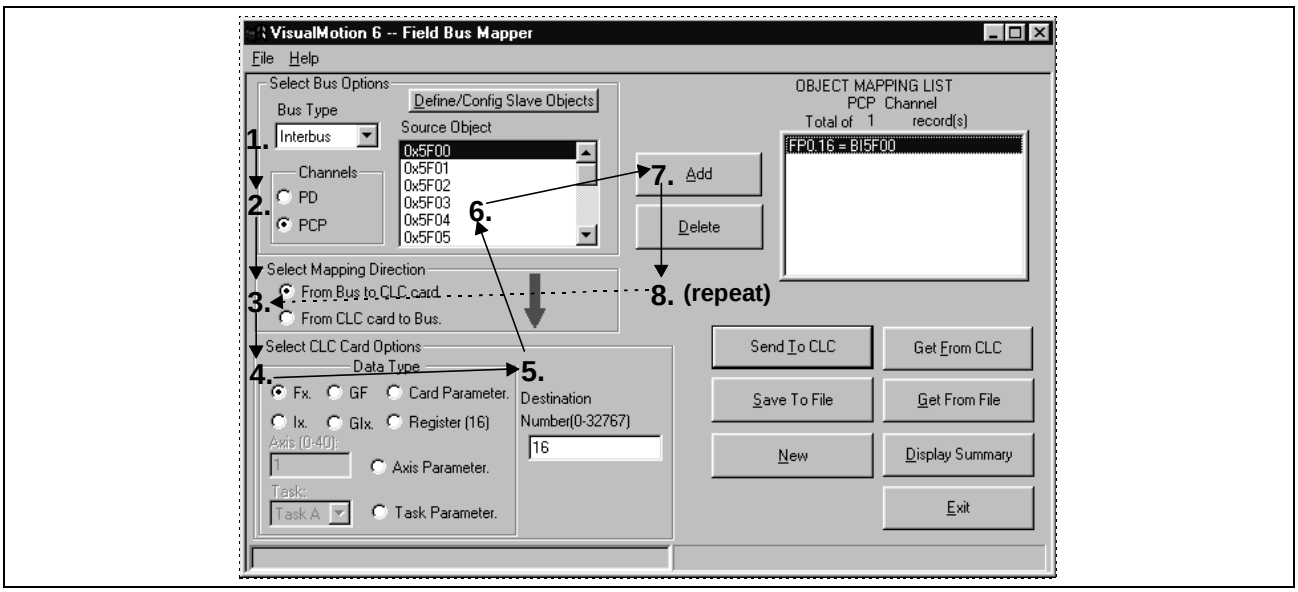


Figure 2-19: Adding Objects to the Non-Cyclic Object Mapping List

9. Click “Send to CLC” to save the selected object mapping list to the CLC-D card. Use any of the following buttons, as needed.

Saves the currently selected object mapping list to the CLC-D Card (to card parameter C-0-2600 if the cyclic [PD] channel is selected, and C-0-2700 if the non-cyclic [PCP] channel is selected).

Saves the currently selected object mapping list to a file (with a .prm extension).

Clears up the current object mapping list.

Gets the currently selected object mapping list from the CLC-D Card (from card parameter C-0-2600 if the cyclic [PD] channel is selected, and C-0-2700 if the non-cyclic [PCP] channel is selected).

Allows the user to open a previously saved file (with a .prm extension) of an object mapping list. When the file is opened, displays that object mapping list.

Generates an on-line summary report of the currently selected Fieldbus. This report can be printed by selecting "Print" from the File menu in the Fieldbus Mapper window.

Exits the Fieldbus Mapper utility.

Note: For debugging purposes, the Fieldbus Mapper follows each mapped item with the cursor as it is saved to the CLC-D. If there is a problem with the mapping of a particular object, an error message appears while the cursor remains on the object.

10. **Mapping is complete!**
For programming information, refer to ***Information for the GPS Programmer*** on page 3-1or ***Information for the PLC Programmer*** on page 4-1.

3 Information for the GPS Programmer

3.1 Viewing/Printing a Summary Report of the Current Fieldbus (only when CLC-D is on-line)

To View a Summary Report:

From the Fieldbus Mapper Window, click on the

button or select "Display Summary" from the File menu.

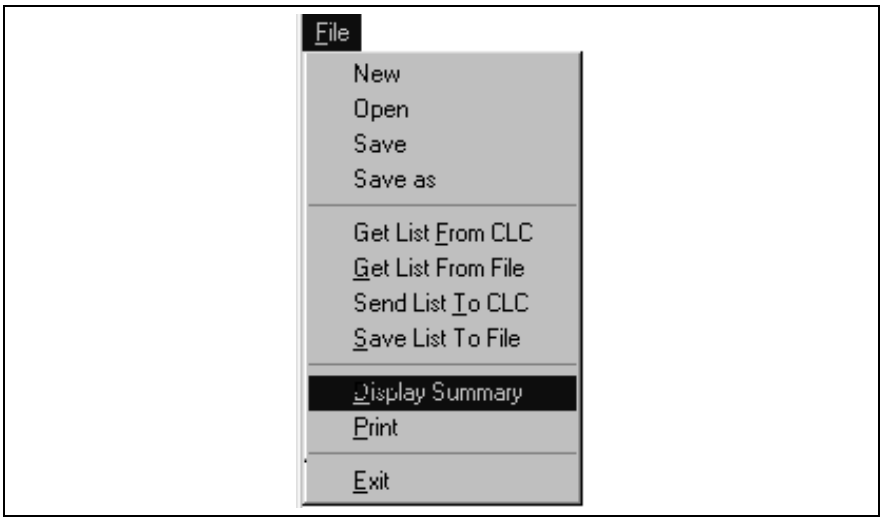


Figure 3-1: File Menu → Display Summary

A scrollable window will appear:

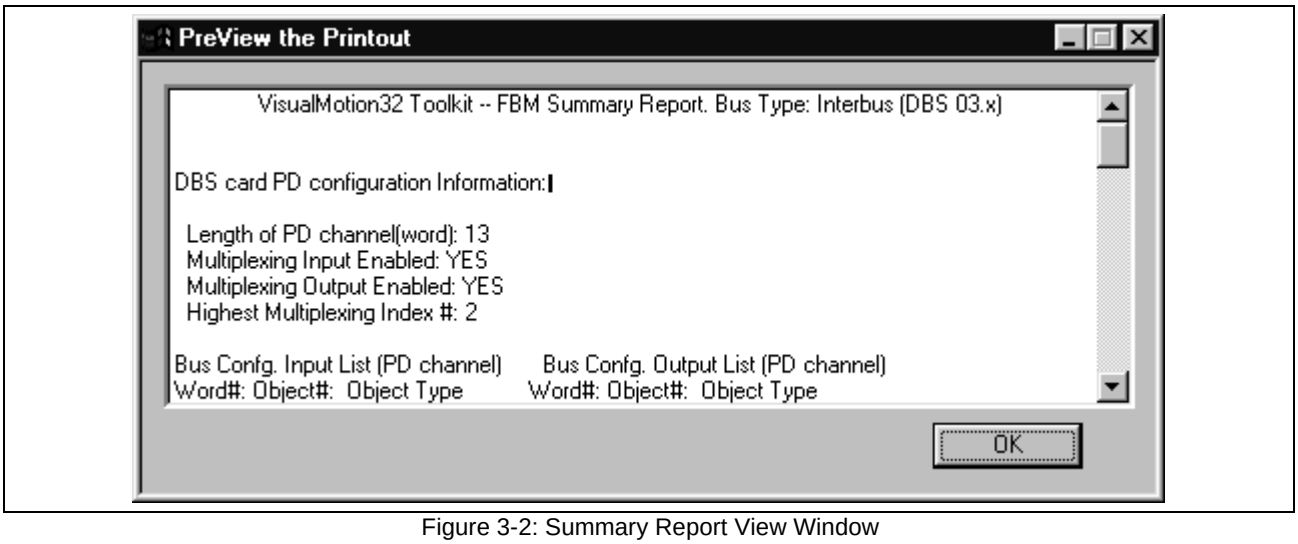

PreView the Printout
VisualMotion32 Toolkit -- FBM Summary Report. Bus Type: Interbus (DBS 03.x)
DBS card PD configuration Information:
Length of PD channel(word): 13
Multiplexing Input Enabled: YES
Multiplexing Output Enabled: YES
Highest Multiplexing Index #: 2
Bus Confg. Input List (PD channel)
Word#: Object#: Object Type
Bus Confg. Output List (PD channel)
Word#: Object#: Object Type
OK

Figure 3-2: Summary Report View Window

To Print a Summary Report:

From the Fieldbus Mapper window, select "Print" from the File menu.

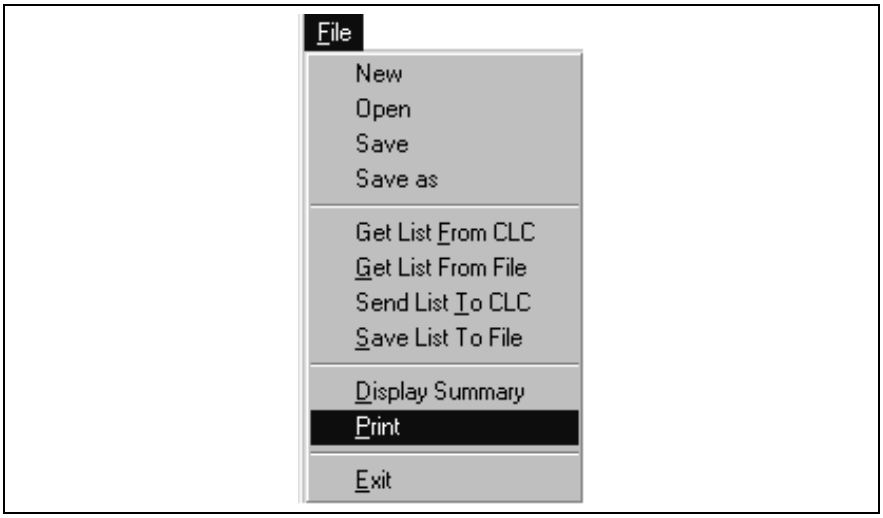


Figure 3-3: File Menu → Print

VisualMotion32 Toolkit – FBM Summary Report. Bus Type: Interbus (DBS 03.x)

Date: 05/05/99 Page 01 Time: 11:18:35

DBS card PD configuration Information:

Device Address: 7
Length of PD channel (word): 13
Multiplexing Input Enabled: YES
Multiplexing Output Enabled: YES
Highest Multiplexing Index #: 2

Bus Confg. Input List (PD channel)

Word#:	Object#:	Object Type
00	0x5F81	16 bit, single
01	0x5EB1	32 bit, single
02	0x0000	.....
03	0x5EB2	32 bit, single
04	0x0000	.....
05	0x5F20	16 bit, baseMP
06	0x5E80	32 bit, baseMP
07	0x0000	.....
08	0x5E90	32 bit, baseMP
09	0x0000	.....
10	0x5EA0	32 bit, base MP
11	0x0000	.....
12	0x5FFd	MP StatusWord

Bus Confg. Output List (PD channel)

Word#:	Object#:	Object Type
00	0x5F91	16 bit, single
01	0x5EF1	32 bit, single
02	0x0000	.....
03	0x5EF2	32 bit, single
04	0x0000	.....
05	0x5F40	16 bit, baseMP
06	0x5EC0	32 bit, baseMP
07	0x0000	.....
08	0x6ED0	32 bit, baseMP
09	0x0000	.....
10	0x5EE0	32 bit, baseMP
11	0x0000	.....
12	0x5FFC	MP ControlWord

CLC/GPS data mapping Lists for DBS card PD channel:

Data mapping IN List (PD channel)

Word#:	Data Type	Object Type
00	RX0.120	16 bit, single
01,02	HP0.22	32 bit, single
03,04	GP0.44	32 bit, single
05	RX0.31	16 bit, mult00
	RX0.32	16 bit, mult01
	RX0.33	16 bit, mult02
06,07	AP1.102	32 bit, mult00
	AP2.102	32 bit, mult01
	AP3.102	32 bit, mult02
08,09	AP1.100	32 bit, mult00
	AP2.100	32 bit, mult01
	AP3.100	32 bit, mult02
10,11	IP0.13	32 bit, mult00
	IP0.23	32 bit, mult01
	IP0.33	32 bit, mult02
12	0x5FFD	MP StatusWord

Data mapping OUT List (PD channel)

Word#:	Data Type	Object Type
00	RX0.100	16 bit, single
01,02	FP0.2	32 bit, single
03,04	IP0.4	32 bit, single
05	RX0.11	16 bit, mult00
	RX0.12	16 bit, mult01
	RX0.13	16 bit, mult02
06,07	FP0.11	32 bit, mult00
	FP0.21	32 bit, mult01
	FP0.31	32 bit, mult02
08,09	IP0.11	32 bit, mult00
	IP0.21	32 bit, mult01
	IP0.31	32 bit, mult02
10,11	IP0.14	32 bit, mult00
	IP0.24	32 bit, mult01
	IP0.34	32 bit, mult02
12	0x5FFC	MP ControlWord

CLC/GPS data mapping Lists for DBS card PCP channel:

Data mapping IN List (PCP channel)

Object#:	Data Type	Object Type
5F10	FP0.5	32 bit
5F11	IP0.8	32 bit

Data mapping OUT List (PCP channel)

Object#:	Data Type	Object Type
5F00	FP0.16	32 bit
5F01	IP0.7	32 bit
5F70	RX0.121	16 bit

Figure 3-4: Printout of Sample Fieldbus Mapping

3.2 Fieldbus-Accessible Parameters

The following tables contain all of the Fieldbus-accessible parameters. Read and write access to these parameters is listed for both cyclic and non-cyclic data.

Parameter		Cyclic		Non-Cyclic (Direct-Mapped)	
Type/No.	Description	Read	Write	Read	Write
A-0-0020	Maximum Velocity	X (3)		X (3)	X (3)
A-0-0021	Maximum Acceleration	X		X	X
A-0-0022	Maximum Deceleration	X		X	X
A-0-0023	Jog Acceleration	X		X	X
A-0-0025	Maximum Jog Increment	X		X	X
A-0-0026	Maximum Jog Velocity	X		X	X
A-0-0031	CLC Cam/Ratio Master Factor (N)	X		X	
A-0-0032	CLC Cam/Ratio Slave Factor (M)	X		X	
A-0-0033	CLC Cam Stretch Factor (H)	X		X	
A-0-0034	CLC Cam Currently Active	X		X	X
A-0-0035	CLC Cam Position Constant (L)	X		X	
A-0-0037	Ratio Mode Step Rate	X		X	X
A-0-0038	Ratio Mode Options	X		X	
A-0-0100	Target Position	X (1, 3)		X (3)	
A-0-0101	Command Position	X (1, 3)		X (3)	
A-0-0102	Feedback Position	X (1, 3)		X (3)	
A-0-0110	Programmed Velocity	X (1, 3)	X (1, 3)	X (3)	X (3)
A-0-0111	Commanded Velocity	X (1, 3)		X (3)	
A-0-0112	Feedback Velocity	X (1, 3)		X (3)	
A-0-0120	Programmed Acceleration	X (1, 3)		X (3)	
A-0-0141	Torque Command	X (1, 3)		X (3)	
A-0-0142	Torque Feedback	X (1, 3)		X (3)	
A-0-0150	Programmed Ratio Adjust	X (1, 3)		X (3)	
A-0-0151	Programmed Phase Offset	X (3)		X (3)	
A-0-0153	CLC Phase Adjust Average Velocity	X (3)		X (3)	X (3)
A-0-0155	CLC Phase Adjust Time Constant	X		X	X
A-0-0157	Current Phase/CLC Cam Master Offset	X (3)		X (3)	
A-0-0159	Ratio Adjust Step Rate	X (3)		X (3)	X (3)
A-0-0160	Commanded Ratio Adjust	X (1, 3)		X (3)	
A-0-0164	ELS Options	X		X	
A-0-0180	Optional Command ID #1	X		X	
A-0-0181	Optional Command ID #2	X		X	
A-0-0182	Optional Command ID #3	X		X	
A-0-0185	Optional Feedback ID #1	X		X	
A-0-0186	Optional Feedback ID #2	X		X	
A-0-0190	Command Data #1	X (1, 2, 3)	X (1, 2, 3)	X (1, 2, 3)	X (1, 2, 3)
A-0-0191	Command Data #2	X (1, 2, 3)	X (1, 2, 3)	X (1, 2, 3)	X (1, 2, 3)
A-0-0192	Command Data #3	X (1, 2, 3)	X (1, 2, 3)	X (1, 2, 3)	X (1, 2, 3)
A-0-0195	Feedback Data #1	X (1, 2, 3)		X (1, 2, 3)	
A-0-0196	Feedback Data #2	X (1, 2, 3)		X (1, 2, 3)	
C-0-0001	Language Selection	X		X	
C-0-0002	Unit Number	X		X	X
C-0-0009	Error Reaction Mode	X		X	
C-0-0010	System Options	X		X	

1 = Must be configured by GPS in SERCOS telegram to update data. Otherwise, unexpected GPS system errors may occur.

2 = The data in this parameter is ONLY interpreted as type Float.

3 = This parameter is valid for fieldbus mapping in firmware version 06v64 and later. The data is read/written via the fieldbus in SERCOS phase 4 only. This parameter requires additional GPS resources to transfer data to/from the fieldbus. Therefore, care must be taken when using it for fieldbus communications with consideration to GPS system resources. Monitor GPS Register 26 for fieldbus mapping resource consumption, and increase the SERCOS update (C-0-0099) if necessary.

4 = No limit-checking is performed when writing to this parameter via a fieldbus. Therefore, care must be taken to ensure the proper data is sent.

Table 3-1: Fieldbus-Accessible Parameters (part 1 of 3)

Parameter		Cyclic		Non-Cyclic (Direct-Mapped)	
Type/No.	Description	Read	Write	Read	Write
C-0-0016	Communication Time-Out Period	X		X	X
C-0-0020	Transmitter Fiber Optic Length	X		X	X
C-0-0021	User Watchdog Timer	X		X	
C-0-0022	User Watchdog Task ID	X	X (4)	X	X
C-0-0030	Option Cards Active	X		X	
C-0-0031	Option Card Status	X		X	
C-0-0042	World Large Increment	X		X	X
C-0-0043	World Small Increment	X		X	X
C-0-0045	World Fast Jog Speed	X		X	X
C-0-0046	World Slow Jog Speed	X		X	X
C-0-0052	Axis Large Increment	X		X	X
C-0-0053	Axis Small Increment	X		X	X
C-0-0055	Axis Fast Jog Velocity	X		X	X
C-0-0056	Axis Slow Jog Velocity	X		X	X
C-0-0090	Download Block Size	X		X	X
C-0-0120	Operating Mode	X		X	
C-0-0121	SERCOS Communication Phase	X		X	
C-0-0123	Diagnostic Code	X		X	
C-0-0125	System Timer Value	X (3)		X	X
C-0-0151	Master 1 Drive Address	X		X	
C-0-0153	Virtual Master Programmed Velocity	X (3)		X (3)	X (3)
C-0-0154	Virtual Master Programmed Acceleration	X (3)		X (3)	X (3)
C-0-0155	Virtual Master Programmed Deceleration	X (3)		X (3)	X (3)
C-0-0156	Virtual Master E-Stop Deceleration	X (3)		X (3)	X (3)
C-0-0157	ELS Master Current Position	X (3)		X (3)	X (3)
C-0-0158	ELS Master Current Velocity	X (3)		X (3)	
C-0-0159	Master 1 Zero Velocity Window	X (3)		X (3)	X (3)
C-0-0160	Virtual Master Max. Jog Velocity	X		X	X
C-0-0161	Master 1 Ratio Input	X		X	
C-0-0162	Master 1 Ratio Output	X		X	
C-0-0164	Master 1 Encoder Type	X		X	
C-0-0170	PLS 1 Mask Register	X		X	X
C-0-0801	Pendant Protection Level 1 Password	X		X	X
C-0-0802	Pendant Protection Level 2 Password	X		X	X
C-0-0803	Pendant User Accessible Floats Section	X		X	X
C-0-0804	Pendant User Accessible Integers Section	X		X	X
C-0-0805	Pendant Start of User Accessible Registers	X		X	X
C-0-0806	Pendant End of User Accessible Registers	X		X	X
C-0-0807	Pendant Password Timeout	X		X	X
C-0-0810	TPT Message and Prompt Control Word	X	X (4)	X	X
C-0-0811	User Task Controlled Menu ID for TPT	X	X (4)	X	X
C-0-0812	User Task Controlled Task ID for TPT	X	X (4)	X	X
C-0-0813	User Task Controlled Axis Number for TPT	X	X (4)	X	X
C-0-0814	TPT Data Transaction Word	X	X (4)	X	X
C-0-1001	ELS Secondary Master	X		X	
C-0-1010	Master Switching Threshold	X (3)		X (3)	X (3)

1 = Must be configured by GPS in SERCOS telegram to update data. Otherwise, unexpected GPS system errors may occur.

2 = The data in this parameter is ONLY interpreted as type Float.

3 = This parameter is valid for fieldbus mapping in firmware version 06v64 and later. The data is read/written via the fieldbus in SERCOS phase 4 only. This parameter requires additional GPS resources to transfer data to/from the fieldbus. Therefore, care must be taken when using it for fieldbus communications with consideration to GPS system resources. Monitor GPS Register 26 for fieldbus mapping resource consumption, and increase the SERCOS update (C-0-0099) if necessary.

4 = No limit-checking is performed when writing to this parameter via a fieldbus. Therefore, care must be taken to ensure the proper data is sent.

Table 3-2: Fieldbus-Accessible Parameters (part 2 of 3)

Parameter		Cyclic		Non-Cyclic (Direct-Mapped)	
Type/No.	Description	Read	Write	Read	Write
C-0-1011	Diff. Between Real Master Positions	X (3)		X (3)	
C-0-1012	Master Synchronization Acceleration	X		X	
C-0-1013	Master Synchronization Time Constant	X		X	
C-0-1014	Master Synchronization Velocity Window	X		X	
C-0-1015	Virtual Master Current Position	X (3)		X (3)	X (3)
C-0-1016	Virtual Master Current Velocity	X (3)		X (3)	
C-0-1017	Position Difference Selection	X		X	X
C-0-1018	Position Monitoring Window	X (3)		X (3)	X (3)
C-0-1019	Position Difference	X (3)		X (3)	
C-0-1508	Master 1 Filter Cutoff Frequency	X		X	
C-0-1509	Master 1 Filter Type	X		X	
C-0-1517	Master 1 Current Position	X (3)		X (3)	X (3)
C-0-1518	Master 1 Current Velocity	X (3)		X (3)	
C-0-1550	Master 2 Type	X		X	
C-0-1551	Master 2 Drive Address	X		X	
C-0-1552	Master 2 Encoder Type	X		X	
C-0-1554	Master 2 Ratio Input	X		X	
C-0-1555	Master 2 Ratio Output	X		X	
C-0-1556	Master 2 Zero Velocity Window	X (3)		X (3)	X (3)
C-0-1558	Master 2 Filter Cutoff Frequency	X		X	
C-0-1559	Master 2 Filter Type	X		X	
C-0-1567	Master 2 Current Position	X (3)		X (3)	X (3)
C-0-1568	Master 2 Current Velocity	X (3)		X (3)	
C-0-2014	Start of SERCOS I-O Station Registers	X		X	
C-0-2015	Registers Allocated per I-O Station	X		X	
T-0-0002	Task Options	X		X	
T-0-0005	World Position Units	X		X	
T-0-0010	Kinematic Number	X		X	
T-0-0011	Coordinated X-Axis	X		X	
T-0-0012	Coordinated Y-Axis	X		X	
T-0-0013	Coordinated Z-Axis	X		X	
T-0-0020	Maximum Path Speed	X (3)		X (3)	
T-0-0021	Maximum Acceleration	X		X	
T-0-0022	Maximum Deceleration	X		X	
T-0-0023	Look Ahead Distance	X		X	X
T-0-0024	Velocity Override	X		X	X
T-0-0025	Maximum Jog Increment	X		X	X
T-0-0026	Maximum Jog Velocity	X		X	X
T-0-0035	Relative Point Used for Origin	X		X	X
T-0-0036	Relative Point Used for Tool Frame	X		X	X
T-0-0100	Target Point Number	X (3)		X (3)	
T-0-0101	Segment Status	X (3)		X (3)	
T-0-0102	Rate Limit Status	X (3)		X (3)	

1 = Must be configured by GPS in SERCOS telegram to update data. Otherwise, unexpected GPS system errors may occur.

2 = The data in this parameter is ONLY interpreted as type Float.

3 = This parameter is valid for fieldbus mapping in firmware version 06v64 and later. The data is read/written via the fieldbus in SERCOS phase 4 only. This parameter requires additional GPS resources to transfer data to/from the fieldbus. Therefore, care must be taken when using it for fieldbus communications with consideration to GPS system resources. Monitor GPS Register 26 for fieldbus mapping resource consumption, and increase the SERCOS update (C-0-0099) if necessary.

4 = No limit-checking is performed when writing to this parameter via a fieldbus. Therefore, care must be taken to ensure the proper data is sent.

Table 3-3: Fieldbus-Accessible Parameters (part 3 of 3)

3.3 Register 19 Definition (Fieldbus Status)

Diagnostic Object 5FF2

The diagnostic object 5ff2 holds the information for "Fieldbus Status," and this information is stored in VisualMotion Register 19. The register information can be referenced in a VisualMotion application program to respond to the status of each bit. The use of these bits is application-dependent.

Table 3-4: Bit Assignment for the Diagnostic Object 5ff2 (VisualMotion Register 19) contains the bit assignment for the diagnostic object 5ff2. The assigned bits are labeled with "x" and the bit number in the second row. Unassigned bits are labeled with "---."

16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
x16	x15	---	x13	---	---	---	---	---	---	x6	x5	x4	---	x2	x1

Table 3-4: Bit Assignment for the Diagnostic Object 5ff2 (VisualMotion Register 19)

Bit Definitions

x1, x2 Status bits for the internal DPR (Dual-Port RAM) communication between the Fieldbus slave and the CLC-D:

x1: FB Init OK , LSB (least significant bit)

x2: FB Init OK, MSB (most significant bit)

The bit combinations for x1 and x2 are as follows:

Bit 2 (CLC-D)	Bit 1 (Fieldbus)	Description
0	0	A reset has been executed on the DPR, **or** neither the CLC-D card nor the Fieldbus card have initialized the DPR.
0	1	The DPR is initialized by the Fieldbus card, but **not** yet by the CLC-D card.
1	0	The DPR initialization is complete. DPR has been initialized by the Fieldbus card and CLC-D card.
1	1	Fieldbus to CLC-D communications system is ready. The system looks for this combination of bit settings to initiate any communication between the Fieldbus and CLC-D cards. The watchdog-function via DPR communications with the Fieldbus slave works correctly.

Table 3-5: Possible Settings for Bits 1 and 2, Status Bits for DPR Communication

x4 Status bit for the active bus capabilities of the Fieldbus slaves (FB Slave Ready)

This bit is monitored for the Fieldbus Error Reaction. Whenever this bit goes to 0 after a Fieldbus card was initially found by the CLC-D, the selected Error Reaction (system shutdown, error message, or ignore) is initiated. See **Fieldbus Error Reaction** on page 3-9 for detailed information about the possible settings.

0--> The Fieldbus slave is not (yet) ready for data exchange.

1--> The Fieldbus slave can actively participate on the bus.

x5 Status bit for the non-cyclic channel (PCP) (Non-Cyc Ready)

0--> The non-cyclic (PCP) channel can not (yet) be used.

1--> The non-cyclic (PCP) channel is ready for use by the Fieldbus Master.

x6 Status-bit for the reconfiguration of the cyclic (PD) channel (nCyc Chan Ready):

0--> The cyclic (PD) channel on the Fieldbus-card is configured properly. The system looks for this bit to be 0 before allowing data transfer.

1--> The cyclic (PD) channel is being reconfigured on the Fieldbus Card at this time.

x13 Status bit for non-supported GPS firmware (nVM FW OK):

0--> The GPS firmware version is supported with this Fieldbus interface.

1--> The GPS-firmware version is NOT supported by this Fieldbus interface card. The CLC-D hardware could support the Fieldbus interface, but the firmware version is not recognized as valid by the Fieldbus card firmware.

x15 Status bit for the cyclic data output (Cyclic Data Valid):

0--> The cyclic data outputs (coming in to the CLC-D) are INVALID.

1--> The cyclic data outputs (coming in to the CLC-D) are VALID. The system looks for this bit to be 1 before allowing data transfer.

x16 Status bit for the multiplex channel (Multiplex Enabled)

0--> The multiplex channel is not (yet) configured or enabled.

1--> The multiplex channel is configured and enabled.

3.4 Register 20 Definition (Fieldbus Diagnostics)

Diagnostic Object 5FF0

The diagnostic object 5ff0 holds the information for "Fieldbus Diagnostics," and this information is stored in VisualMotion Register 20.

Table 3-6: Bit Assignment for the Diagnostic Object 5ff0 (VisualMotion Register 20) contains the bit assignment for the diagnostic object 5ff0. The assigned bits are labeled with "x" and the bit number in the second row. Unassigned bits are labeled with "---."

16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
---	x15	x14	x13	---	---	---	---	x8	x7	x6	x5	x4	x3	x2	x1

Table 3-6: Bit Assignment for the Diagnostic Object 5ff0 (VisualMotion Register 20)

Bit Definitions

x1 - x8 Fieldbus Interface-specific fault code (FB Card Fault Code)

Note: This function is currently not available.

x13 - x15 Identification of the Fieldbus interface card (FB Card Found)

The bit combinations for x13, x14 and x15 are as follows:

Bit 15	Bit 14	Bit 13	Fieldbus Type
0	0	0	<NO CARD>
0	0	1	<Not Defined>
0	1	0	Interbus (DBS03)
0	1	1	DeviceNet (DCF01)
1	0	0	Profibus (DPF05)
1	0	1	CanOpen (DCF01)
1	1	0	<Not Defined>
1	1	1	<Not Defined>

Table 3-7: Identification of the Fieldbus Interface

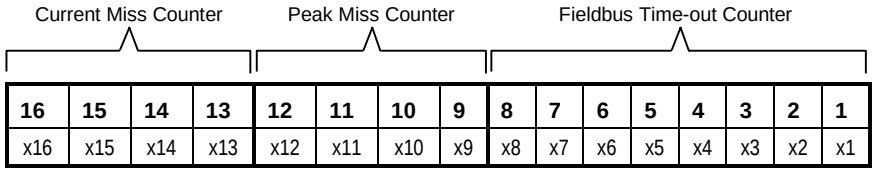
3.5 Register 26 Definition (Fieldbus Resource Monitor)

The "Fieldbus Resource Monitor" in register 26 can be used as a method for monitoring the attempts made to process the Object Mapping List parameter C-0-2600 across the Dual-port RAM. If after three SERCOS update cycles, the Object Mapping List is not successfully transmitted, a "**miss**" is noted.

Register 26 is divided into the following three counter types:

- Current Miss Counter.
- Peak Miss Counter.
- Fieldbus Timeout Counter.

Table 3-8: Bit Assignment for Fieldbus Resource Monitor (Register 26) contains the bit assignment for the Fieldbus counters. The assigned bits are labeled with an "x" followed by the bit number.



16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
x16	x15	x14	x13	x12	x11	x10	x9	x8	x7	x6	x5	x4	x3	x2	x1

Table 3-8: Bit Assignment for Fieldbus Resource Monitor (Register 26)

Note: View register 26 in Hexadecimal format to more easily monitor the Fieldbus counters.

Hex display format of register 26

0x0000

Fieldbus Time-out Counter (00-FF)
Increments by one every time 10 cyclic mapping list attempts are missed in 10 attempts.

Peak Miss Counter (0-9)
Displays the peak value of the Current Miss Counter.

Current Miss Counter (0-9)
Displays the count of missed cyclic mapping list attempts out of the last 10 attempts

Note: To view registers in hex, select ***Data ⇒ Registers*** within VisualMotion Toolkit. If the registers are not currently viewed in hex format, select ***Format ⇒ Hex*** in the *Active Program, Register* window.

Bit Definitions

x13 - x16 Status bits for the "Current Miss Counter."

An attempt is made to transmit the cyclic Object Mapping List, C-0-2600, across the Dual-port RAM every third SERCOS update cycle. For every 10 mapping list update attempts (30 SERCOS update cycles), the failed attempts are counted and displayed in these bits (values can range from 0-9). If 10 out of 10 mapping list updates attempts are missed, the "Fieldbus Timeout Counter" is incremented by one. This is an indication of a Fieldbus Mapping Timeout Error.

x9 - x12 Status bits for the "Peak Miss Counter."

These bits monitor the "Current Miss Counter's" peak count between a value from 0-9 and holds that value until a larger count is encountered.

x1 - x8 Status bits for the "Fieldbus Timeout Counter."

The count of these bits increments by one every time the "Current Miss Counter" encounters 10 out 10 missed attempts of the cyclic mapping list update. A count incremented by one represents a Fieldbus Mapping Timeout Error and is process by GPS according to the selected "Fieldbus Error Reaction" in parameter C-0-2635. See Error! Reference source not found. for an explanation of the Fieldbus Error Reaction setting.

Note: The GPS programmer can monitor the "Current Miss Counter" and define a custom error reaction for missed mapping list update attempts less than 10.

Note: The values in register 26 are read/write and can be reset by the user.

3.6 Fieldbus Error Reaction

Note: The Fieldbus Error Reaction setting is active only in SERCOS Phase 4. In all other SERCOS phases, it will be inactive.

You can choose how you would like the CLC-D system to react in case of a Fieldbus error. This reaction can be set in the "Configure Slave Telegram" screen, using the combo box labeled "Fieldbus Error Reaction."

Three options are available for the Error Reaction setting. Depending on the selected setting, the value 0, 1, or 2 is stored in Parameter C-0-2635:

Setting	Value in Parameter C-0-2635
Shutdown CLC	0 (default)
Warning Only	1
Ignore	2

Table 3-9: Parameter C-0-2635 Values for Error Reaction Settings

Fieldbus Mapper Timeout The Fieldbus Mapper continually scans the system for sufficient resources to process the cyclic data mapping list (2600-list). If 10 mapping list updates are missed in a row, the system is considered to

have insufficient resources and the selected error reaction is evoked, as follows:

If "Shutdown CLC" (0) is set in Parameter C-0-2635, the following error is generated from the CLC-D card: **520 Fieldbus Mapper Timeout**

If "Warning Only" (1) is set in Parameter C-0-2635, the following error is generated: **209 Fieldbus Mapper Timeout**

If "Ignore" (2) is set in Parameter C-0-2635, the system will update as resources become available, but there is no way to monitor whether or not updates actually occur.

Lost Fieldbus Connection

Object 5ff2, the dual-port RAM location on the Fieldbus card, indicates the status of the Fieldbus. Register 19 Bit 4, the Fieldbus Slave Ready Bit on the CLC card, mirrors object 5ff2. See ***3.3 Register 19 Definition (Fieldbus Status)*** for more specific bit information. The system monitors this bit and evokes the selected error reaction if the bit is low (0), after a Fieldbus card is found. A typical situation that will cause this condition is the disconnection of the Fieldbus cable from the DBS card.

If "Shutdown CLC" (0) is set in Parameter C-0-2635, the following error is generated from the CLC-D card (active in SERCOS Phase 4 only): **519 Lost Fieldbus Connection**

If "Warning Only" (1) is set in Parameter C-0-2635, the following error is generated (active in SERCOS Phase 4 only): **208 Lost Fieldbus Connection**

If "Ignore" (2) is set in Parameter C-0-2635, there is no noticeable reaction when Register 19 Bit 4 goes low, unless the GPS coding is customized to evoke a special reaction.

Troubleshooting Tip:

If a Fieldbus card is not found on the system, the Error Reaction setting will be ignored. If you have a Fieldbus card and the Error Reaction is not responding as expected, the system may not "see" your Fieldbus card.

4 Information for the PLC Programmer

4.1 Slave Configuration

The Indramat DBS cards support auto-configuration Interbus functionality. No configuration information is necessary.

4.2 Multiplex Data Bits in the Control and Status Words

Figure 4-1: Bit Definitions in Multiplex Control and Status Words contains diagrams of the bits contained in the control and status words and their bit definitions. The bits marked with ✖ are currently not used. The control word is appended as the last word of the cyclic data content, and the status word is appended as the slave response in the cyclic data input.

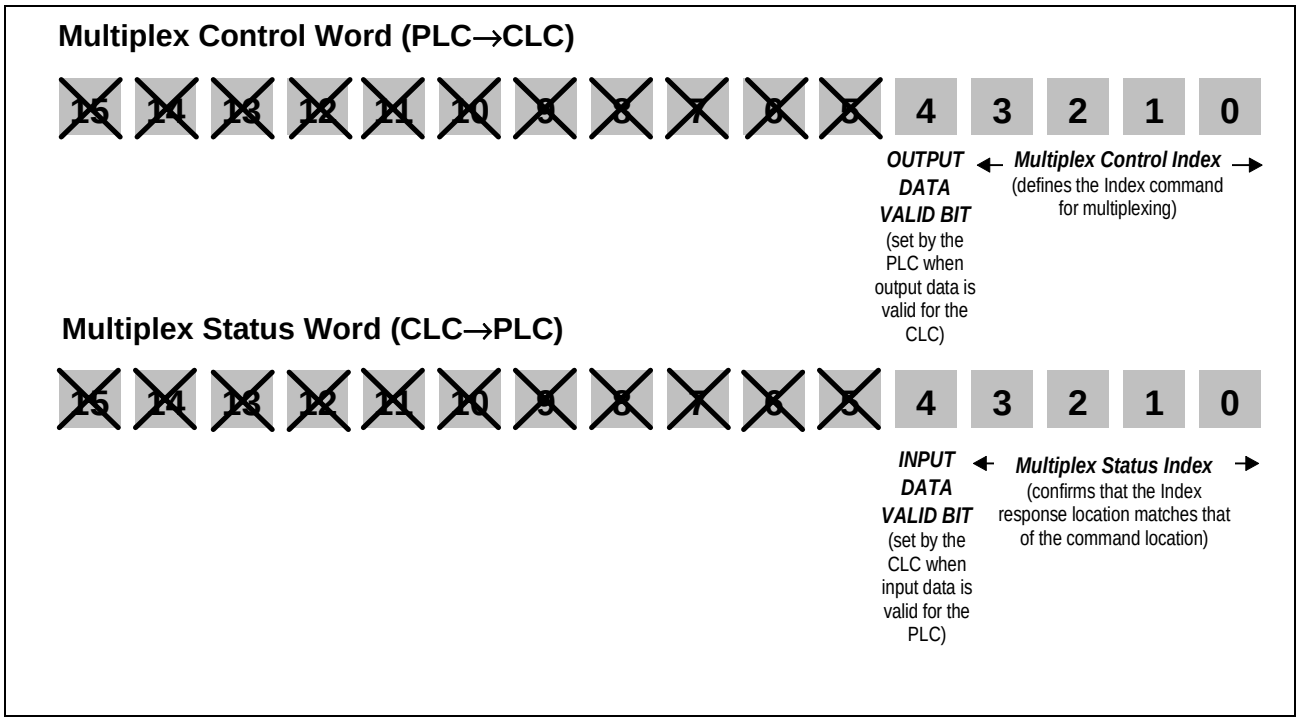

Multiplex Control Word (PLC→CLC)
15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
OUTPUT DATA VALID BIT (set by the PLC when output data is valid for the CLC)
Multiplex Control Index (defines the Index command for multiplexing)
Multiplex Status Word (CLC→PLC)
15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
INPUT DATA VALID BIT (set by the CLC when input data is valid for the PLC)
Multiplex Status Index (confirms that the Index response location matches that of the command location)

Figure 4-1: Bit Definitions in Multiplex Control and Status Words

Figure 4-2: Write / Write and Read Process and ***Figure 4-3: Read-Only Process*** explain the exchange of multiplex information between the master (PLC) and the slave (CLC-D) using the control and status words:

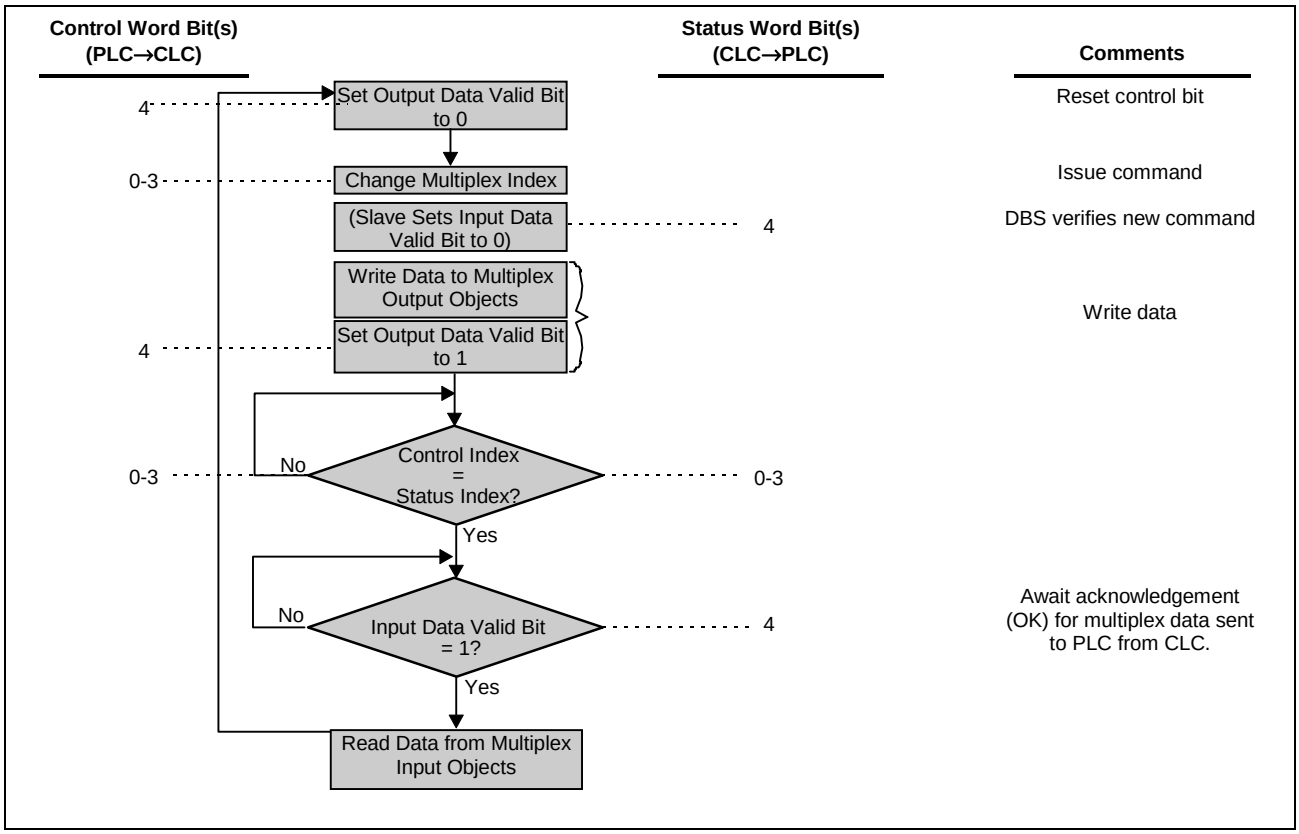


Figure 4-2: Write / Write and Read Process

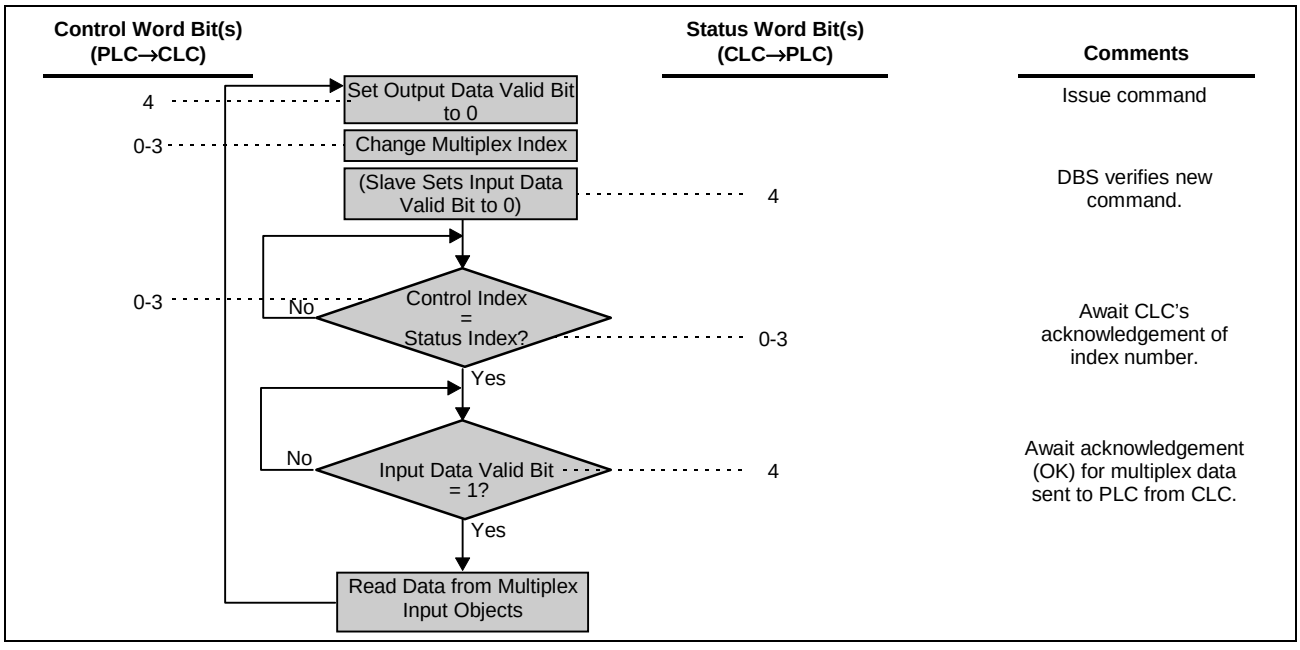


Figure 4-3: Read-Only Process

4.3 Non-Cyclic Transmission (Direct-Mapped Objects)

A number of objects (see ***Table 4-10: Non-Cyclic Data Objects*** below) can represent VisualMotion data types if they are mapped in the non-cyclic object mapping list (C-0-2700). This method provides a simple but inflexible method of transferring non-cyclic data from the master to the slave.

Number of Objects Available	Data Size	Direction	Numeric Names of Objects
16	32-bit	in	5F10-5F1F
16	32-bit	out	5F00-5F0F
32	16-bit	in	5F60-5F6F, 5FA0-5FAF
32	16-bit	out	5F70-5F7F, 5FB0-5FBF

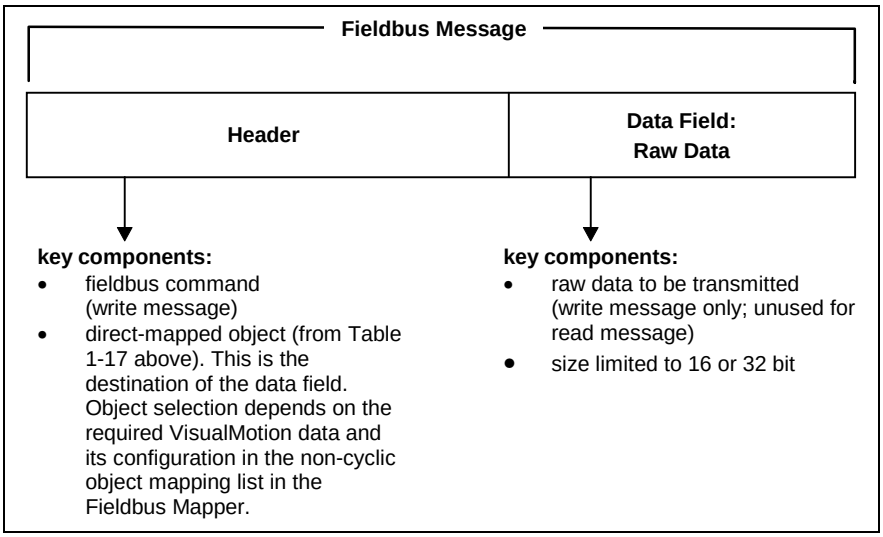
Table 4-10: Non-Cyclic Data Objects

Selecting a Direct-Mapped Object

The VisualMotion data available for non-cyclic direct-mapped objects can be viewed and printed via the summary report function in the VisualMotion Fieldbus Mapper tool (see ***Viewing/Printing a Summary Report of the Current Fieldbus*** on page 3-1).

Transmission Sequence via a Direct-Mapped Object

Note: For the direct-mapped object, only one transmission (and one response) is required, to send a read or write message to the CLC-D card and to receive a response from the CLC-D card.


Fieldbus Message
Header
Data Field:
Raw Data
key components:
• fieldbus command (write message)
• direct-mapped object (from Table 1-17 above). This is the destination of the data field. Object selection depends on the required VisualMotion data and its configuration in the non-cyclic object mapping list in the Fieldbus Mapper.
key components:
• raw data to be transmitted (write message only; unused for read message)
• size limited to 16 or 32 bit

Important: The format of the Fieldbus message header and the method of implementation are dependent on the Fieldbus type and the master (PLC) being used. Refer to your Fieldbus master/PLC documentation for proper transport and formatting of the message header.

Non-Cyclic Direct-Mapped Write

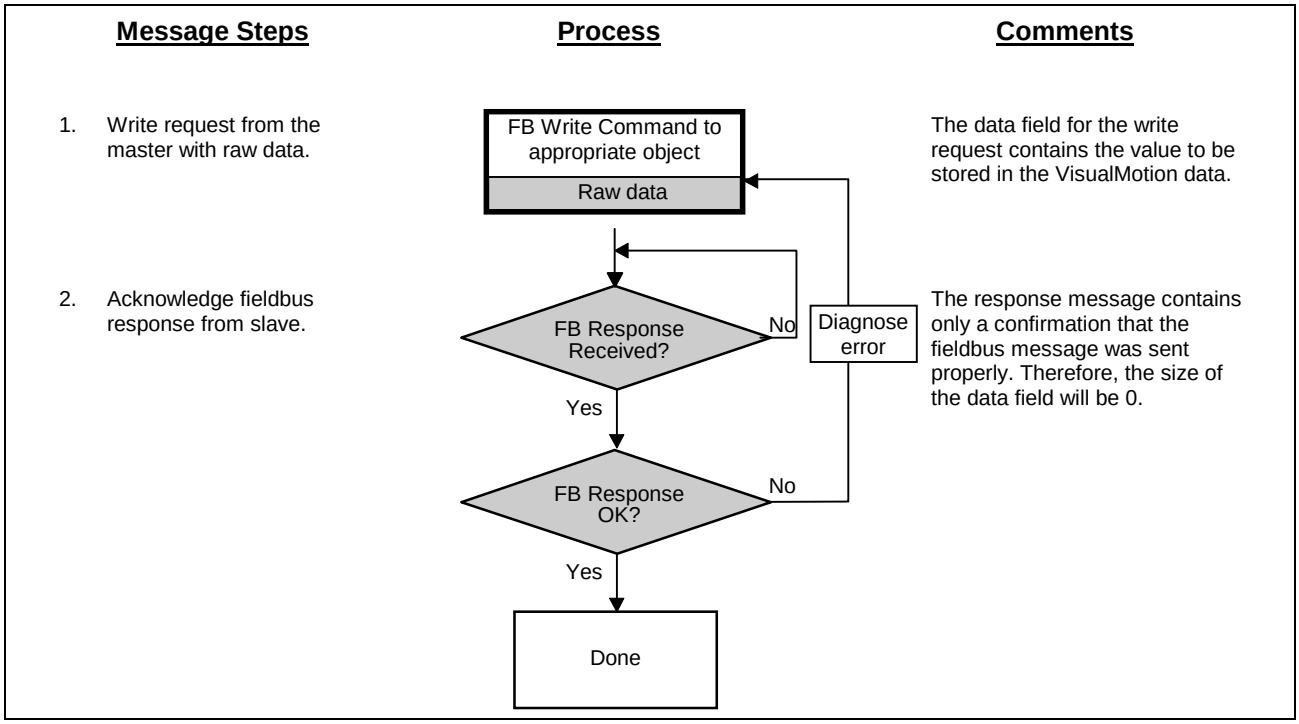
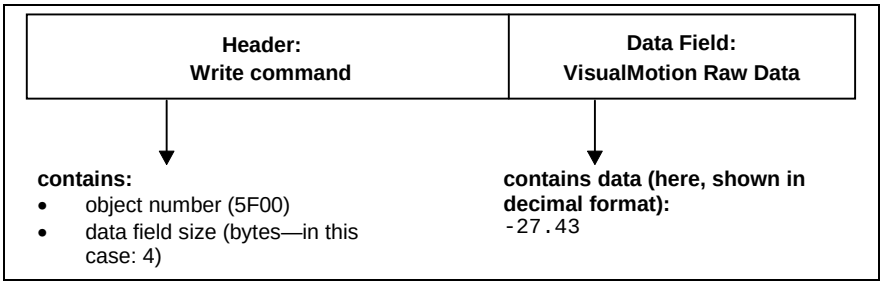

Message Steps
Process
Comments
1. Write request from the master with raw data.
FB Write Command to appropriate object
Raw data
The data field for the write request contains the value to be stored in the VisualMotion data.
2. Acknowledge fieldbus response from slave.
FB Response Received?
No
Diagnose error
The response message contains only a confirmation that the fieldbus message was sent properly. Therefore, the size of the data field will be 0.
Yes
FB Response OK?
No
Yes
Done

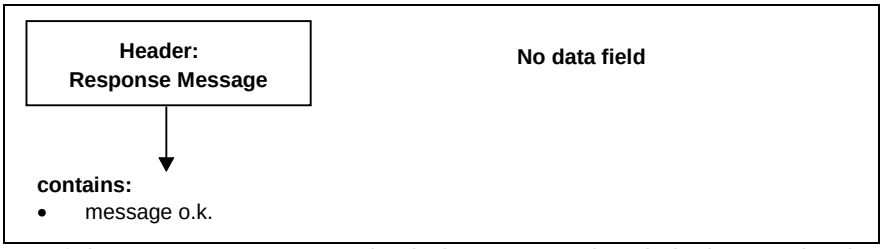
Figure 4-4: Non-Cyclic Direct-Mapped Write Process

Example: Write the value -27.43 to Float 16 (This is a 32-bit non-cyclic output object. In our Fieldbus Mapper example, it is mapped to object 5F00.)

1. Write request from the master with raw data.



2. After the write request from the master, the CLC-D card sends a response message.



3. If the message response (code in message header) shows o.k., the transaction is complete.

Note: It is important to note the mapping of the object in the Fieldbus Mapper. If a particular object is mapped as a “VisualMotion Read Object,” you **cannot write** to that object. Conversely, if the object is mapped as a "VisualMotion Write Object," you **cannot read** from that object.

Non-Cyclic Direct-Mapped Read

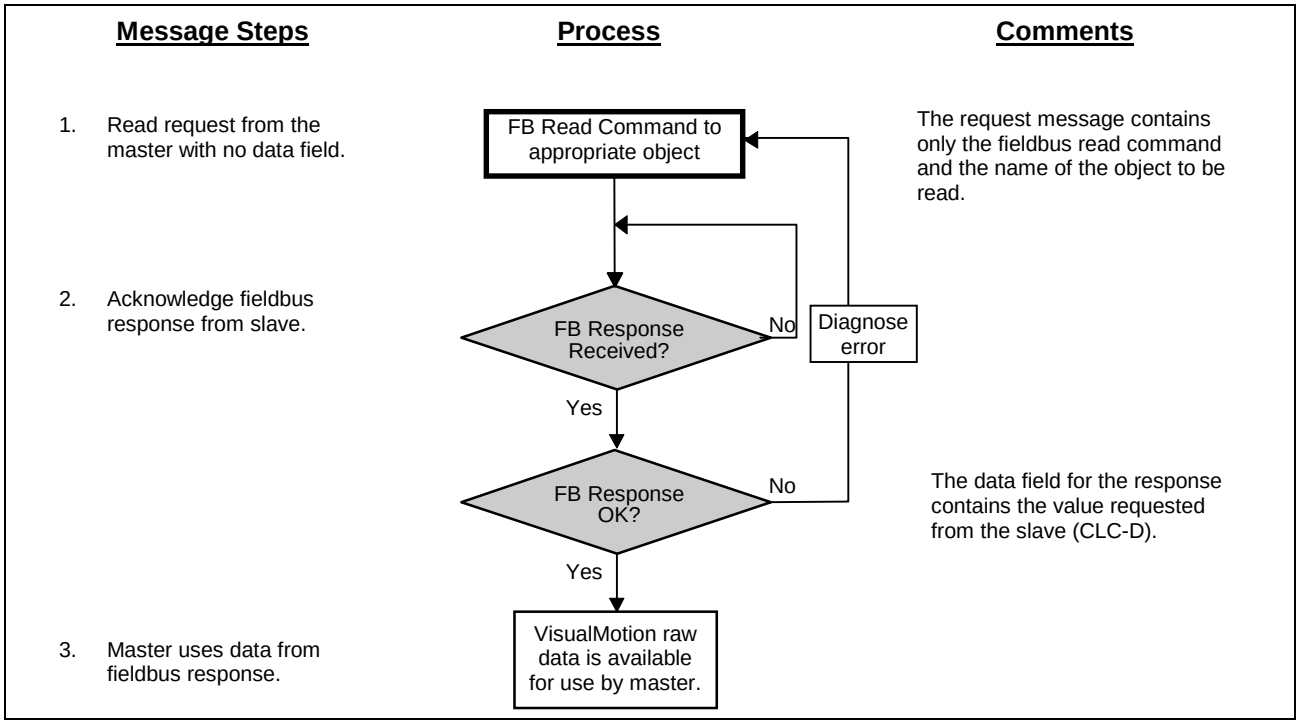
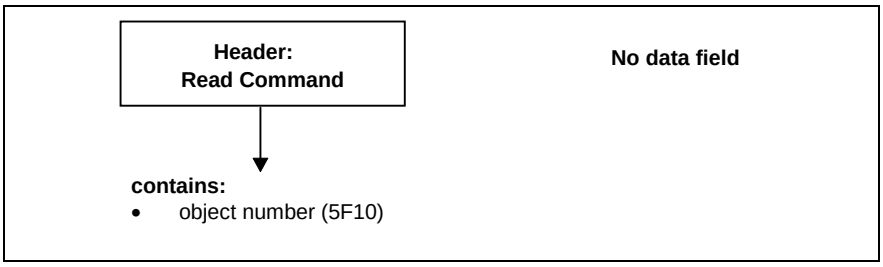

Message Steps
Process
Comments
1. Read request from the master with no data field.
FB Read Command to appropriate object
The request message contains only the fieldbus read command and the name of the object to be read.
2. Acknowledge fieldbus response from slave.
FB Response Received?
No
Diagnose error
Yes
FB Response OK?
No
The data field for the response contains the value requested from the slave (CLC-D).
Yes
3. Master uses data from fieldbus response.
VisualMotion raw data is available for use by master.

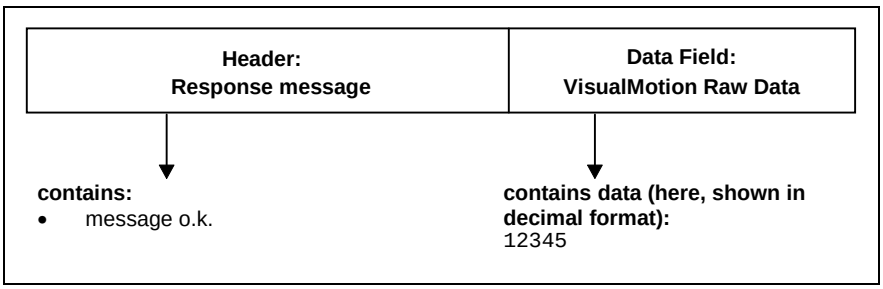
Figure 4-5: Non-Cyclic Direct-Mapped Read Process

Example: Read the value contained in Integer 8. (This is a 32-bit non-cyclic input object. In our Fieldbus Mapper example, it is mapped to object 5F10.)

1. Read request from the master.



2. After the read request from the master, the CLC-D card sends a response message.



If the message response (code in message header) shows o.k., the requested value is attached to the message in the data field. This value is now available for use by the master (PLC).

Note: It is important to note the mapping of the object in the Fieldbus Mapper. If a particular object is mapped as a “VisualMotion Read Object,” you **cannot write** to that object. Conversely, if the object is mapped as a "VisualMotion Write Object," you **cannot read** from that object.

4.4 Non-Cyclic Transmission (Data Exchange Objects)

The four data exchange objects 5E70 to 5E73 represent fixed data "containers" of varying lengths that transfer the VisualMotion ASCII Protocol to the CLC-D card. These objects serve as an open-ended possibility to access any VisualMotion data (including cams, diagnostic text, etc.), but more work is required in the master to perform a transmission of this type. Both the VisualMotion ASCII message and the Fieldbus transfer message must be formulated.

Table 4-1: Length of the Data Exchange Objects lists the available data exchange objects and their sizes.

Data Exchange Object	Data Length (in bytes)
5E70	16
5E71	32
5E72	64
5E73	128

Table 4-1: Length of the Data Exchange Objects

Selecting a Data Exchange Object

Depending on the length of a VisualMotion ASCII message, any of these data exchange objects can be selected.

Note: The entire data length of the data exchange object must always be transmitted even if the VisualMotion ASCII message is shorter.
For example, if you want to transmit an ASCII message of 42 bytes, you must use object 5E72. To avoid a response error from the Fieldbus slave, you must append 22 "Null" characters to the end of the ASCII message to complete a data size of 64 bytes.

Note: The checksum for the VisualMotion ASCII protocol is NOT used with the data exchange object. If the checksum is sent as part of the string, it will be ignored, and no checksum will be sent in the VisualMotion ASCII response messages. To ensure data integrity, the Fieldbus protocols support a low-level checksum.

Transmission Sequence via a Data Exchange Object

Note: For the data exchange object, two transmissions (and two responses) are required, to send the read or write message to and then receive the response message from the CLC-D card.

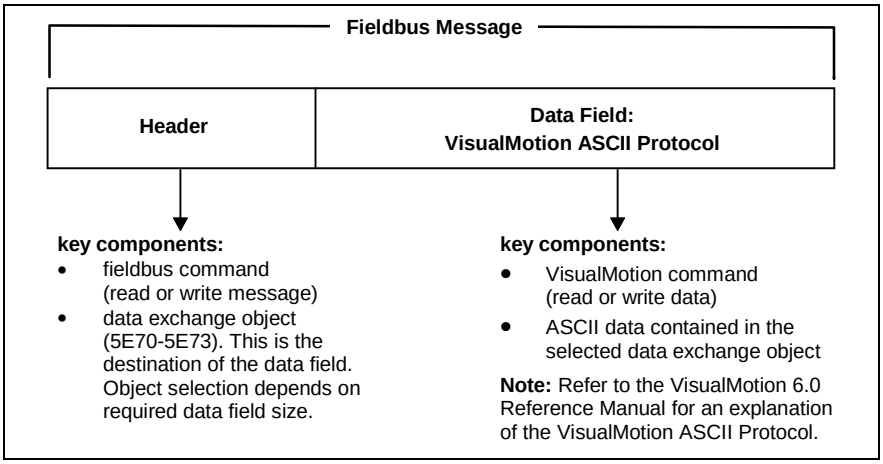

Fieldbus Message
Header
Data Field:
VisualMotion ASCII Protocol
key components:
• fieldbus command (read or write message)
• data exchange object (5E70-5E73). This is the destination of the data field. Object selection depends on required data field size.
key components:
• VisualMotion command (read or write data)
• ASCII data contained in the selected data exchange object
Note: Refer to the VisualMotion 6.0 Reference Manual for an explanation of the VisualMotion ASCII Protocol.

Figure 4-6: Format of a Non-Cyclic Fieldbus Message using a Data Exchange Object

Important: The format of the Fieldbus message header is dependent on the type of master (PLC) being used. Refer to your PLC manufacturer's manual for specific information on this topic.

The following sequence describes the communication between the Fieldbus master (PLC) and the Fieldbus slave (CLC-D):

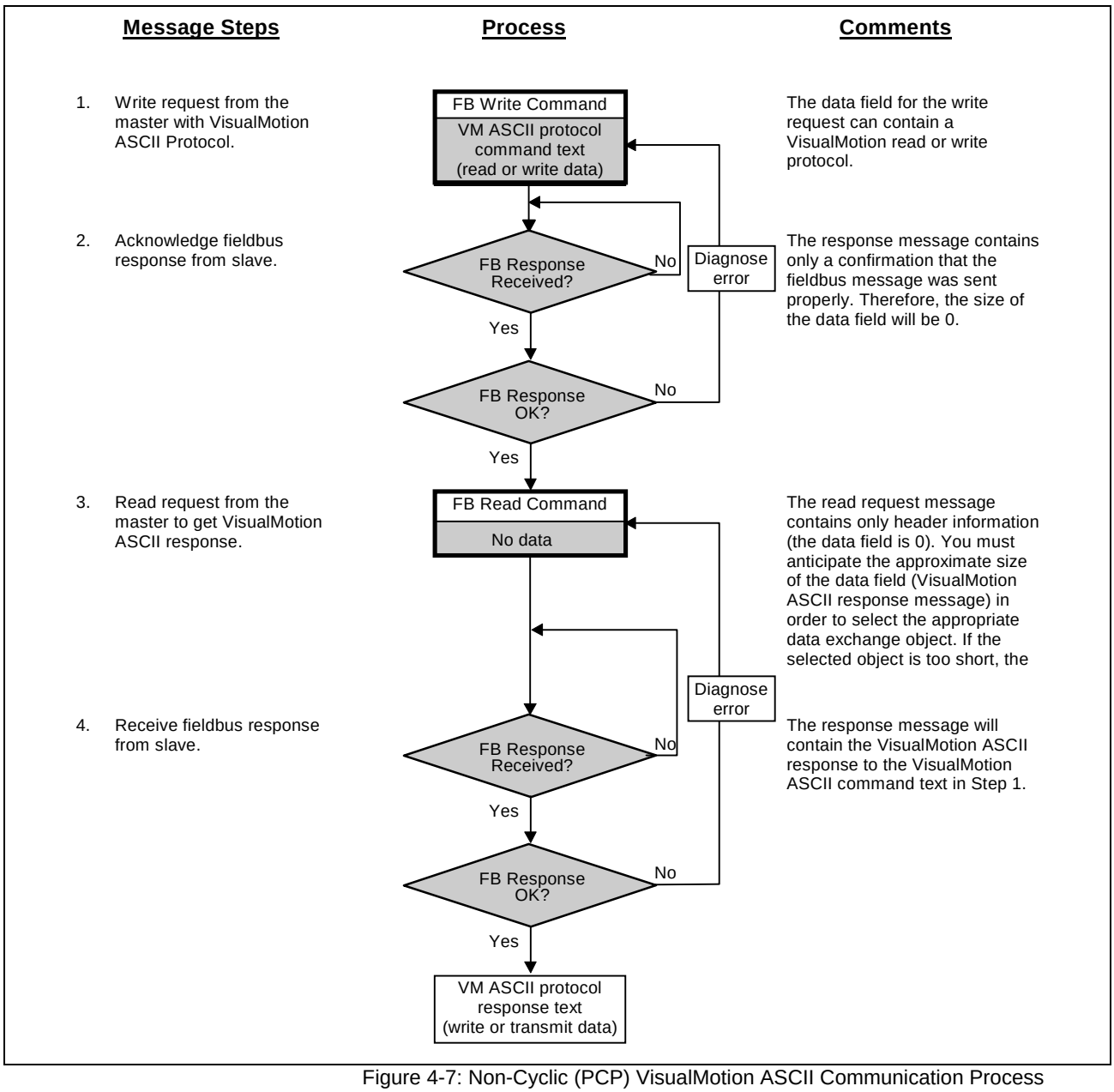
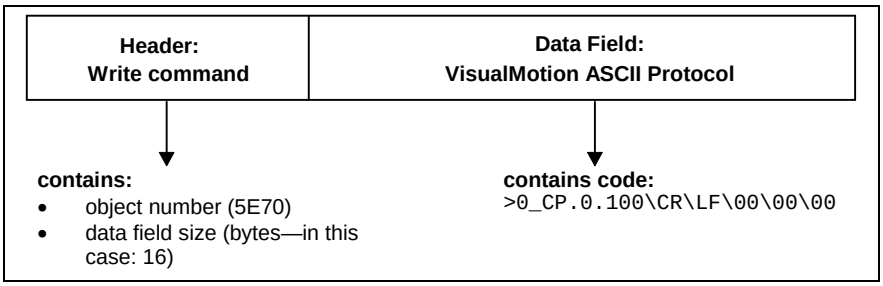

Message Steps
Process
Comments
1. Write request from the master with VisualMotion ASCII Protocol.
FB Write Command
VM ASCII protocol command text (read or write data)
The data field for the write request can contain a VisualMotion read or write protocol.
2. Acknowledge fieldbus response from slave.
FB Response Received?
No
Diagnose error
Yes
FB Response OK?
No
Yes
The response message contains only a confirmation that the fieldbus message was sent properly. Therefore, the size of the data field will be 0.
3. Read request from the master to get VisualMotion ASCII response.
FB Read Command
No data
The read request message contains only header information (the data field is 0). You must anticipate the approximate size of the data field (VisualMotion ASCII response message) in order to select the appropriate data exchange object. If the selected object is too short, the
4. Receive fieldbus response from slave.
Diagnose error
FB Response Received?
No
Yes
FB Response OK?
No
Yes
The response message will contain the VisualMotion ASCII response to the VisualMotion ASCII command text in Step 1.
VM ASCII protocol response text (write or transmit data)

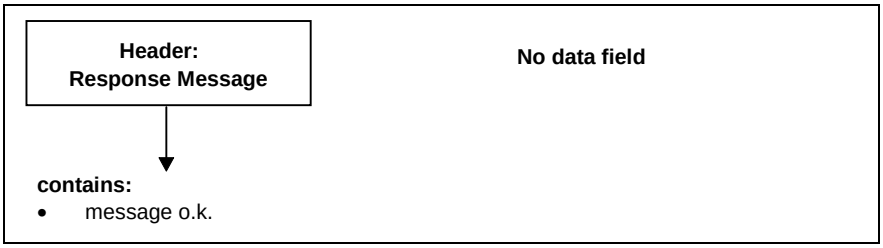
Figure 4-7: Non-Cyclic (PCP) VisualMotion ASCII Communication Process

Example: Read Card Parameter 100 (CLC-D firmware version)

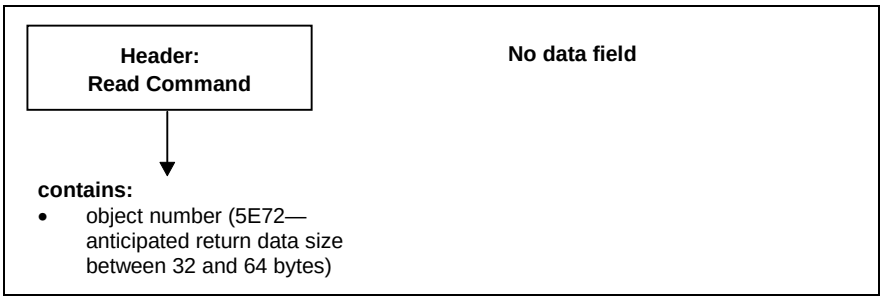
1. Write request from the master with VisualMotion ASCII Protocol.



2. After the first read request from the master, the CLC-D card sends a response message.

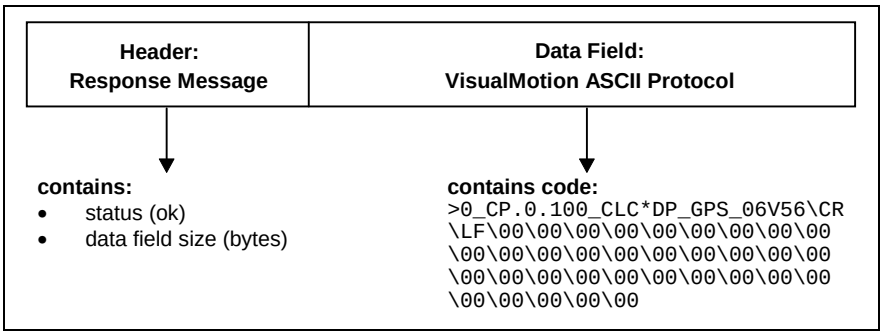


3. Read request from the master for the VisualMotion ASCII response message.



Note: To ensure that all of the data requested in this step is received in step 4 below, a data exchange object of the appropriate size must be selected.
If the selected data exchange object is too small, the data will be truncated.
If the selected data exchange object is too large, efficiency of transmission will be compromised.

4. The CLC-D sends the final response message.



5 Interbus Slave Board DBS03.x

5.1 Application

The DBS03.x Interbus Slave Card enables the inclusion of the CLC-D control card in an Interbus system. The Interbus card supports the real-time cyclic channel (Process Data--PD) with up to 16 words and enables non-cyclic data exchange between master and slave via the non-cyclic (Peripherals Communication Protocol--PCP) Interbus channel.

With this hardware, CLC-D parameters and data sets, as well as parameters of an attached drive controller can be transmitted.

For the highest possible flexibility, the cyclic and non-cyclic channels are freely configurable by the user when using the object structure of the DBS03.x card.

5.2 Functionality

The DBS03.x card has the following characteristics:

- Interbus Slave with support of PCPS 2.0.
- Long-distance bus-to-interface connection with completely isolated interfaces, according to Interbus DIN # 19258 specifications.
- Freely configurarable cyclic channel with 1-16 word capacity on the bus.
- Two D-SUB connectors for the incoming and outgoing bus, according to and described in DIN specification 19258:
  -X40incoming bus9-pin D-SUB male connector
  -X41outgoing bus9-pin, D-SUB female connector
- Data exchange to the CLC-D control module via Dual-Port Memory.
- 8 LEDs integrated in the Front-Panel, with diagnostic field according to Interbus specification.
- Implementation of an object structure to simplify access to variables and parameters in the control card and drives.
- Upload / Download function via 4 arrays, from 16 to 128 Bytes (Data Exchange Object).
- DBS04.1 Board ONLY: Power-fail functionality, which prevents the Interbus network from going down when the DBS/CLC loses power.

5.3 Bus Configuration

Each device on the Interbus network, including the DBS slave card, reports its information (ID, words, etc.) to the Interbus master.

The Indramat DBS03.x supports automatic configuration of the bus, which functions as described in the Interbus specification.

5.4 DBS03.x Board Hardware

Front view of the DBS03.x

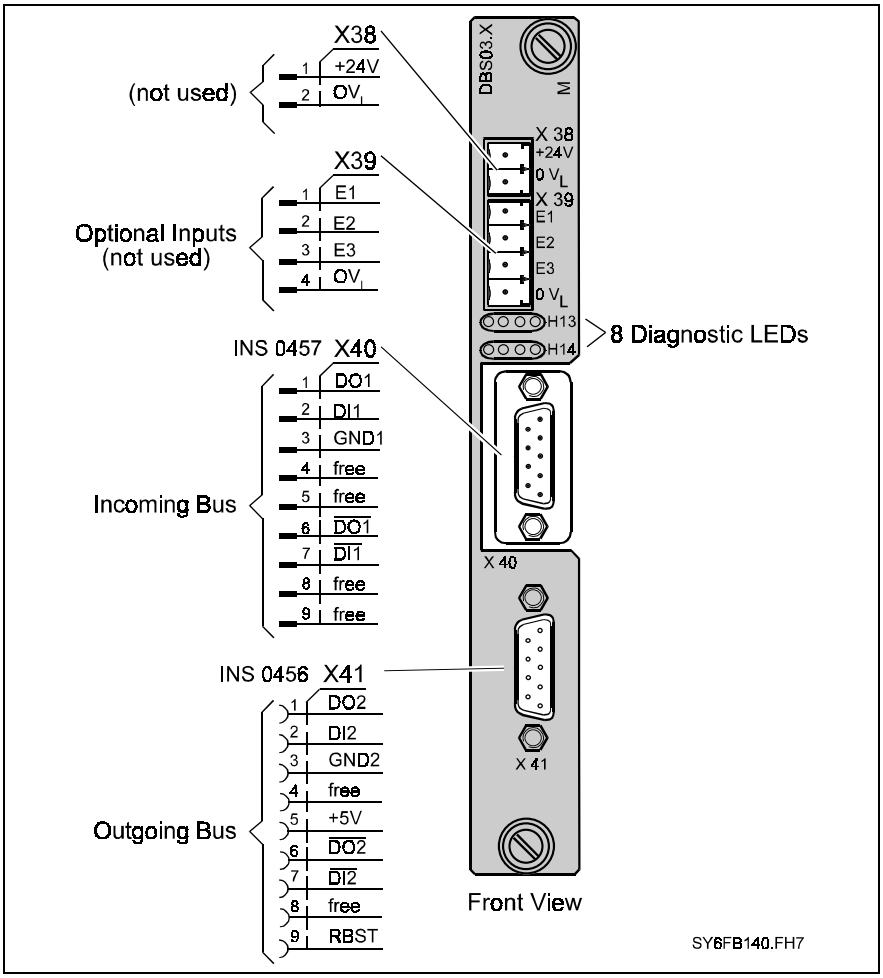


Figure 5-1: Front View of the DBS03.x

DBS03.x Structure

Note: Other cards can be plugged into the DBS03.x. Please take this into consideration when disassembling or removing the DBS card from the card cage.

Power (+5V) is supplied by the drive or CCD card cage via another connector on the back of the card. However, signals are always transmitted via the plug-in connection to the CLC-D.

The interface to the CLC-D is achieved via an Indramat-specific extended bus. However, only one DBS03.x card can be connected to the CLC-D.

Note: The DBS03.x card cannot be used with a DPF05.x Profibus slave card or a DCF01.x DeviceNet slave card.

The DBS03.x card has the following interfaces:

Interface to the Drive or CCD Card Cage

required only for power supply to the DBS03.x if power is not supplied by the CLC-D.

Interface to the CLC-D or other Cards

Information is transmitted to the CLC-D and any other cards via this interface. It is an extended bus interface with partial coding of the addressing field, so that additional cards can be connected (e.g. DEA28, DEA29, DEA30).

External Inputs

The DBS03.x card provides three hardware inputs (+24V). These inputs can only be used with the CLC-D if the appropriate firmware is available. The signal status at these inputs are transmitted to the CLC-D independently of the Interbus Status (On/Off), but can also be requested from the Interbus Master via the PD or PCP channels (these external inputs are currently not supported).

External Power

The DBS03.x is supplied with power internally. However, no power-fail function exists if that internal power fails. If the power-fail functionality is desired, the DBS04.x must be used.

Interbus Interface

The Interbus interface is designed according to DIN specification 19258 and is optically isolated. Connection can be made via a 9-pin D-SUB connector.

X40 Connector, Interbus Pin-outs, Incoming Bus

X40	Signal	Name
1	DO 1	Out RS-485
2	DI 1	In RS-485
3	GND 1	Ground
4	---	not used
5	---	not used
6	/DO 1	/Out RS-485
7	/DI 1	/In RS-485
8	---	not used
9	---	not used

Table 5-1: X40 Connector, Interbus Pin-outs, Incoming Bus

X41 Connector, Interbus Pin-outs, Outgoing Bus

X41	Signal	Name
1	DO 2	Out RS-485
2	DI 2	In RS-485
3	GND 2	Ground
4	---	not used
5	+5V 2	+5 Volts
6	/DO 2	/Out RS-485
7	/DI 2	/In RS-485
8	RBST	Remote bus control
9	---	not used

Table 5-1: X41 Connector, Interbus Pin-outs, Outgoing

X39 Connector, External Inputs

Note: Not functional at this time.

X39	Description	Input Voltage, High	Input Voltage, Low
1	E1	+16 V... +32 V	-0.5 V ... +8 V
2	E2	+16 V... +32 V	-0.5 V ... +8 V
3	E3	+16 V... +32 V	-0.5 V ... +8 V
4	E4	Reference Potential, 0V	Reference Potential, 0V

Table 5-2: X69 Signal Configuration, External Inputs

DBS03.x Diagnostics

Front Panel LEDs

The DBS03.x front panel has eight diagnostic LEDs. They enable the diagnosis of the Interbus current status and the communication between the DBS03.x and CLC-D cards.

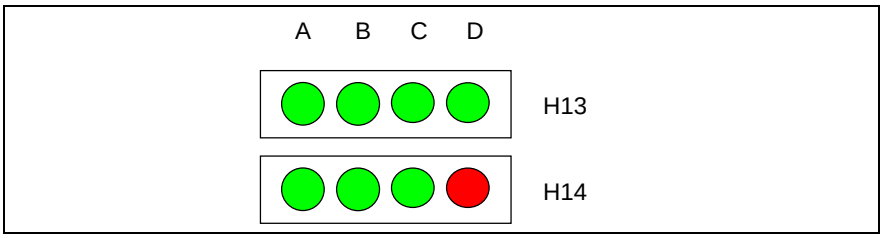


Figure 5-2: Arrangement of DBS03.x Board Diagnostic LEDs

Definition of Diagnostic LEDs

LED	Signal	Status	Definition
H13A	**UL--Power source of Interbus board o.k.**	Green	This LED is on when the Interbus slave receives +5V. This voltage is taken from the DC/DC converter on the DBS03.x board and made available to the optically isolated bus drivers.
H13B	**BA--Interbus PD active**	Green	This LED is on if the Interbus cyclic channel (PD) is active. The cyclic data is continually monitored. If data transfer is detected during a time-out, this LED turns on.
H13C	**TR--PCP transmission active**	Green	This LED is on while a non-cyclic PCP communication is executed on the Interbus ring. Initialization of the PCP channel and selection of the object structure (Get OV) are also considered PCP communications.
H13D	**RC--Remote Check**	Green	With this LED, the bus between the current device and the adjacent (previous) device is checked. If the connection is o.k., the LED turns on. Upon initializing the Interbus ring, the LED turns on only if the entire ring is o.k.
H14A	**SW-RUN--Software-RUN**	Green	This LED is used with LED H14C. It supports diagnosis of correctly running software and successful synchronization between the CLC-D and DBS03.x cards. LEDs H14A and H14C indicate, by their flashing frequency, the current SERCOS cycle time. If the CLC-D and DBS03.x cards are synchronized, the LEDs flash alternately. If the two LEDs flash in unison, there is a synchronization problem, although the Interbus is functioning on this board.

H14B	**+24V—+24V External Voltage**	Green	This LED is used only for the DBS04.1 board.
H14C	**SW-RUN-- Software-RUN**	Green	See description for LED H14A
H14D	**RD—Remote Bus Disable**	Red	This LED is on when the continuing bus is not o.k. The board recognizes whether or not a continuing bus is connected via the RBST signal, which is connected to +5V via a jumper. If a continuing cable is connected, but the connection to the next device is interrupted, this LED turns on when the Master completes a bus check.

Table 5-3: LED Definitions

5.5 Interbus Process Data (Cyclic Channel)

DBS03.x Process Data Configuration

The DBS03.x card is an intelligent Interbus card that can be configured according to process requirements. Not only does it provide a particular number of words which are transmitted in both data directions during each Interbus cycle, but it also allows the user to determine which data word is placed in which location in the data segment.

This flexibility enables assigning data to addressable elements, or objects, that are designated with index numbers for different Fieldbuses. The user can then choose the required objects from the configuration stored on the CLC-D board, and place them in the desired order on the bus.

This configuration possibility requires the support of the PCP channel by the master.

If the user cannot configure the cyclic channel, configuration can be done using a PC, a Phoenix-Contact Interbus card (e.g. PC-ATT or PC-ISA) and the software tools provided by Phoenix-Contact.

Each time the DBS03.x Interbus card is powered up with the CLC-D card, the bus configuration information is transmitted from the CLC-D to the DBS03.x.

Monitoring the DBS03.x Process Data

Data transfer on the Interbus is generally categorized as secure. If data is transferred from the master to the slaves, it is subjected to extensive CRC checks, before it is accepted as valid data and activated by the slave. Data which is recognized as erroneous (HD = 4, for Interbus), is not forwarded to the CLC-D. Of course, this is also true for the data direction to the master.

Watchdog Function: The DBS03.x card provides bus monitoring by using a watchdog timer, so that an appropriate reaction can be effected when the bus is down. This function uses the object 6003 (PD Monitoring Time). The time (in ms) can be set, for the case of an Error Reaction upon bus failure.

Error Reaction Upon Bus Failure: Two possible error reactions can be set for the Process Data channel, using the object 6004 (Process Data Channel Error Reaction):

- **Retain Output Data**
  If a value of 0x0000 is written to object 6004, the output data is retained upon bus failure. The data that was last output is the current data. A bus failure is reported to the CLC-D's status register.
- **Reset Output Data**
  For critical applications, and especially in cases where movement is controlled by output signals, resetting the data makes sense.
  If a value of 0x0001 is written to object 6004, the output data is reset upon bus failure.

5.6 Interbus Peripherals Communication Protocol (Non-Cyclic Channel)

Supported PCP Services for the DBS03.x

The following PCP actions are supported by the DBS03.x card:

- INITIATE
  PCP connection to a device via the Master.
- ABORT
  Cancellation, by the Master, of a connection with a designated device.
- STATUS
  The device status, which contains information about the current operating and communication status, is read.
- IDENTIFY
  Identification data of the device (Virtual Field Device—VFD) is read. This data contains:
  - Vendor Name
  - Device/Mode Name
  - Revision Number
- GET-OV
  The user can read the object directory of the DBS03.x card and receives information about the implemented objects and their structure. Two GET-OV types are supported:
  - GET-OV long
  - GET-OV short
- REJECT
  Invalid services are rejected by the slave. Invalid services are usually those which are not implemented or do not allow access to objects.
- READ
  Read access to an object in the object directory. Any object in this directory can be read, even if it is used in the Process Data Channel.
- WRITE
  Master data can be written asynchronously to the Process data Channel. Write access is only allowed to those objects that are not used in the Process Data Channel. Others will be rejected using the REJECT action.

The DBS03.x card supports all mandatory actions according to DIN 19258.

DBS03.x PCP Firmware

The DBS03.x card supports the PCP functions of software revision 2.0. Therefore, if is fully compatible with the functions and diagnostics of the new G4 master activation.

By forcing the PCP Channel to a data width of one word, this card is fully backward-compatible to the PCP 1.51 software and is therefore suitable for a master activation of group G3.

Note: It is recommended to use a G4 card for a master activation, in order to optimize the functionality of the DBS03.x.

Communication via the PCP 2.0 software is the basis of the client-server model. A call-answer protocol is used, that is initiated by the client and answered by the server. The DBS03.x can be considered a server, in this case.

The PCP 2.0 software supports the functions specified by the Peripheral Message Specification (PMS). These are subsets of the PCP Interbus specification (DIN 19256 T2), which is in turn part of the Manufacturing Message Specification (MMS).

Addressing of PCP Devices on the Bus

Devices and cards in the Interbus ring are normally addressed according to the position of each device. In an IDENTIFY cycle during INITIALIZATION of the bus, the master determines how many devices with which bus length are on the Interbus ring. The master can recognize which data word belongs to which device, because all devices are seen as a large array.

If devices have PCP capabilities, an additional addressing system is necessary for these devices. The addressing of the Process Data is not affected.

In Interbus, the communication between a master and a slave begins/ends between Level 7 of the OSI model and the actual application. These end points are designated with internal address that is known as the "Communication Reference" (KR) by Interbus and other Fieldbus systems.

Only the PCP-compatible devices receive a KR designation, beginning with KR 2. Non-PCP-compatible devices that are located between PCP-compatible devices do not receive a KR designation.

Note: The DBS03.x card supports **one** PCP access at a time. Only one KR is available, and **no parallel services** are supported.

Setting Interbus device addresses via switches, jumpers or software settings is not possible, because of the addressing according to location on the bus and the allocation of KR designations as addresses for PCP-compatible devices.

5.7 DBS03.x Objects

Objects for Diagnosing and Configuring the DBS03.x

Name	Object Number	Type	Access from Master	PD Channel	Notes
Dummy Object	5FF1	u16	R/RW	Yes	Used as a filler object for the second word in double-word objects.
CLC-D Operating Mode	5FF4	u16	R	Yes	
Fieldbus Operating Mode	5FF3	u16	R	Yes	
Fieldbus Diagnostic	5FF0	u16	R	Yes	
Fieldbus Status	5FF2	u16	R	Yes	
CLC-D Status	5FF5	u16	R	Yes	
CLC-D Error Code	5FF6	u16	R	Yes	
CLC-D Control Word	5FF7	u16	R/RW	Yes	
CLC-D Address / Axis Number	5FF8	u16	R	Yes	
Reserve	5FF9	u16	R/RW	Yes	
PD Monitoring Time	6003	u16	R/RW	No	
Feldbus Control Word	5FFB	u16	R/RW	Yes	
Multiplex Control Word	5FFC	u16	R/RW	Yes	
Multiplex Status Word	5FFD	u16	R	Yes	
Multiplex Channel Start-Offset	5FFE	u16	R/RW	No	First location in the list pointing to the beginning of multiplex data.

Table 5-4:Objects for Diagnosing and Configuring the DBS03.x

Diagnostic Array 5E7A

Name:	Object Number	Type	Access from Master	PD Channel	Notes
CLC-D Error Code	5FF0.1	u16	R/RW	No	(5FF6)
CLC-D Status	5FF0.2	u16	R	No	(5FF5)
Fieldbus Diagnostic	5FF0.3	u16	R	No	(5FF0)

Table 5-5: Diagnostic Array Object 5FF0

6 Index

*

*.gsd file....4-1
Basic Example 2-3
Multiplex Example 2-14

2

208 Lost Fieldbus Connection..3-10
209 Field Bus Mapper Timeout.3-10

5

519 Lost Fieldbus Connection..3-10
520 Field Bus Mapper Timeout.3-10
5E70.... 4-19
5E71.... 4-19
5E72.... 4-19
5E73.... 4-19

A

Available Objects
Basic Example 2-4
Multiplex Example
Cyclic Data....2-14

B

baud rates
supported by DPF05.x.... 5-1
Bit Definitions
Register 19
Cyclic Data Valid....3-7
FB Init OK 3-6
FB Slave Ready 3-6
nCyc Chan Ready 3-7
Non-Cyc Ready....3-7
nVM FW OK....3-7
Register 20
FB Card Fault Code 3-8
FB Card Found 3-8
Register 26
Current Miss Counter 3-9
Fieldbus Timeout Counter....3-9
Peak Miss Counter....3-9
Bus Conf. IN List
Basic Example 2-3
Multiplex Example
Cyclic Data....2-14
Bus Conf. OUT List
Basic Example.... 2-5
Multiplex Example
Cyclic Data 2-15
Bus Configuration Lists
APPEND Button....2-4, 2-14
DEL Button....2-4, 2-14
INSERT Button....2-4, 2-14
NEW Button 2-4, 2-14
Bus Type
Basic Example.... 2-2
Cyclic Data 2-6
Multiplex Example
Cyclic Data 2-12, 2-16
Non-Cyclic Data.... 2-18
Buttons
APPEND 2-4, 2-14
DEL 2-4, 2-14
Display Summary....2-7, 2-9, 2-18, 2-19
Exit....2-7, 2-9, 2-18, 2-19
Get From CLC.....2-7, 2-9, 2-18, 2-19
Get From File 2-7, 2-9, 2-18, 2-19
INSERT 2-4, 2-14
NEW..... 2-4, 2-7, 2-9, 2-14, 2-18, 2-19
Save To File 2-7, 2-9, 2-18, 2-19
Send To CLC.....2-7, 2-9, 2-18, 2-19

C

C-0-2600.....2-7, 2-9, 2-18, 2-19
C-0-2600....1-2
C-0-2635....3-9
C-0-2700.....1-7, 2-9, 2-18, 2-19
channels
Cyclic Data Channel configuration1-3, 1-6
multiplex 1-2
Non-Cyclic Channel.... 1-2
Parameter Channel 1-2
Real-Time Channel 1-2, 1-3
single 1-2
Clearing Parameter Channel Errors4-4
Command Telegram Structure
Short Format 2 4-7
VisualMotion ASCII Format.... 4-11
Configure Multiplex Channel
Basic Example.... 2-3
control word 1-5
Multiplex Example
Cyclic Data 2-14
Cyclic Channel....1-2
Cyclic Data Channel
Basic Example.... 2-6

configuration ... 1-3, 1-6
Multiplex Example ... 2-16
Cyclic Data Valid ... 3-7

D

Data Configuration List
Param ... 2-20
Data Exchange Objects ... **1-8**, 4-19
5E70 ... 4-19
5E70 to 5E73 ... 1-8
5E71 ... 4-19
5E72 ... 4-19
5E73 ... 4-19
Data Mapping
Direct-Mapped Data ... **1-7**
Data Objects ... **1-7**
Data Sizes
Cyclic ... 1-4
Data Type
Basic Example
Cyclic Data ... 2-6
Non-Cyclic Data ... 2-8
Multiplex Example
Cyclic Data ... 2-16
Non-Cyclic Data ... 2-18
Data Types
Cyclic ... 1-4
Multiplex ... 1-3, 1-4, 2-11
Single ... 1-3, 1-4
Define/Config Slave Objects
Basic Example ... 2-3
Multiplex Example
Cyclic Data ... 2-13
Destination
Basic Example
Cyclic Data ... 2-6
Non-Cyclic Data ... 2-8
Multiplex Example
Cyclic Data ... 2-16
Non-Cyclic Data ... 2-18
Device Address
Basic Example ... 2-3
Multiplex Example
Cyclic Data ... 2-13
Diagnostics
LEDs ... 5-3
Direct-Mapped Data ... **1-7**
Display Summary
Cyclic, Multiplex Data ... 2-18
Cyclic, Single Data ... 2-7
File Menu entry ... 3-1
Non-Cyclic, Multiplex Data ... 2-19
Non-Cyclic, Single Data ... 2-9
DPF05.x
Hardware ... 5-2
Interfaces ... 5-2
DPF05.x board ... 1-1

E

EMC safety ... 5-1
Error Bit (E) ... **4-4**
Error Reaction
Ignore ... 3-10
Shutdown CLC ... 3-10
Warning Only ... 3-10
Errors
208 Lost Fieldbus Connection ... 3-10
209 Field Bus Mapper Timeout ... 3-10
519 Lost Fieldbus Connection ... 3-10
520 Field Bus Mapper Timeout ... 3-10
Exit
Cyclic, Multiplex Data ... 2-18
Cyclic, Single Data ... 2-7
Non-Cyclic, Multiplex Data ... 2-19
Non-Cyclic, Single Data ... 2-9
External Inputs ... 5-3

F

FB Card Fault Code ... 3-8
FB Card Found ... 3-8
FB Init OK ... 3-6
FB Slave Ready ... 3-6
Field Bus Error Reaction
Basic Example ... 2-3
Multiplex Example
Cyclic Data ... 2-14
Field Bus Mapper
Examples
Multiplexing ... 2-11
Parameter Channel ... 2-20
Examples
Basic ... 2-1
Fieldbus Diagnostics ... 3-7
Fieldbus Error Reaction ... 3-9
Fieldbus Mapper ... 1-1
Fieldbus Master ... 1-1
fieldbus message header ... 4-16
Fieldbus Resource Monitor ... 3-8
Fieldbus Slave ... 1-1
Fieldbus Status ... 3-6
fieldbus summary report ... 2-7
Fieldbus-Accessible Parameters 3-3
File Menu

Display Summary ... 3-1
Print ... 3-2
first cycle ... **4-4**
FMS (Fieldbus Message Specification) ... 1-2
From FieldBus
Basic Example ... 2-4
Multiplex Example
Cyclic Data ... 2-15

G

Get From CLC
Cyclic, Multiplex Data ... 2-18
Cyclic, Single Data ... 2-7
Non-Cyclic, Multiplex Data ... 2-19
Non-Cyclic, Single Data ... 2-9
Get From File
Cyclic, Multiplex Data ... 2-18
Cyclic, Single Data ... 2-7
Non-Cyclic, Multiplex Data ... 2-19
Non-Cyclic, Single Data ... 2-9
GPS Programming ... 3-1

H

H15A ... 5-4
H15B ... 5-4
H15C ... 5-4
H15D ... 5-4
H16A ... 5-4
H16B ... 5-4
H16C ... 5-4

I

Ignore (Fieldbus Error Reaction) 3-10
Index Number
Multiplex Example
Cyclic Data ... 2-14
input ... 1-2

L

last cycle ... **4-4**
least significant bit ... 3-6
LEDs
H15A ... 5-4
H15B ... 5-4
H15C ... 5-4
H15D ... 5-4
H16A ... 5-4
H16B ... 5-4
H16C ... 5-4
LEDs (DPF05.x) ... 5-3
Length of DP Channel (Word)
Basic Example ... 2-3
Multiplex Example
Cyclic Data ... 2-13

M

Mapping Direction
Basic Example
Cyclic Data ... 2-6
Non-Cyclic Data ... 2-8
Multiplex Example
Cyclic Data ... 2-16
Non-Cyclic Data ... 2-18
Mapping List
Back To Add Button ... 2-8
Delete Button ... 2-8
Insert Button ... 2-8
Replace Button ... 2-8
Messaging Examples
Short Format 2 ... **4-8**, 4-13
Messaging Formats
Short Format 2 ... **1-6**
VisualMotion ASCII Format .. **1-6**, **4-11**
most significant bit ... 3-6
Multiplex Data Bits ... ***4-1***
Multiplex Data Types . 1-3, 1-4, 2-11
Multiplexing
Control and Status Words ... **1-5**
Enable ... 2-14

N

nCyc Chan Ready ... 3-7
New
Cyclic, Multiplex Data ... 2-18
Cyclic, Single data ... 2-7
Non-Cyclic, Multiplex Data ... 2-19
Non-Cyclic, Single Data ... 2-9
Non-Cyc Ready ... 3-7
Non-Cyclic Channel ... 1-2, **1-7**
Non-Cyclic Data Channel
Multiplex Example ... 2-18
Non-Cyclic Data Mapping Lists
Direct Mapping
Basic Example ... 2-8
Multiplex Example ... **2-18**
Non-Cyclic Transmission
Data Exchange Objects ... 4-19
Number of Valid Data Bytes (#Bytes) ... **4-5**
nVM FW OK ... 3-7

O

OBJECT MAPPING LIST
- Basic Example
  - Cyclic Data....2-6
  - Non-Cyclic Data....2-9
- Multiplex Example
  - Cyclic Data....2-17
  - Non-Cyclic Data....2-18

OK, Send to CLC
- Basic Example....2-5
- Multiplex Example
  - Cyclic Data....2-15

output....1-2

P

Parameter Channel ... 1-2, 1-6, 2-20
- Short Format 2....1-6
- VisualMotion ASCII Format....1-6

Parameter Mode
- Basic Example....2-7
- Multiplex Example....2-18

Parameters....3-3

P-CHAN Components
- Control/Status Word....4-4
- **Error Bit (E)....4-4**
- **first cycle....4-4**
- **last cycle....4-4**
- **Number of Valid Data Bytes (#Bytes)4-5**
- **Toggle Bit (T)....4-5**
- **Transmission Format (Fmt)....4-5**

Pin-outs (DPF05.x)....5-3

PLC Programming....4-1
- ***Multiplex Data Bits....4-1***
- Non-Cyclic Data (via Data Exchange Objects)....4-19
- Non-Cyclic Data (via Direct Mapping)4-16
- Parameter Channel....4-3

Print
- File Menu entry....3-2

Print a Summary Report....3-2

Profibus DP Combi Slave Board.5-1

R

Real-Time Channel....1-2, 1-3

Register 19 Definition (Fieldbus Status)....3-6

Register 20 Definition (Fieldbus Diagnostics)....3-7

Register 26 Definition (Fieldbus Resource Monitor)....3-8

Response Telegram Structure
- Short Format 2....4-7
- VisualMotion ASCII Format....4-12

S

Save to File
- Cyclic, Multiplex Data....2-18
- Cyclic, Single Data....2-7
- Non-Cyclic, Multiplex Data....2-19
- Non-Cyclic, Single Data....2-9

Send To CLC
- Cyclic, Multiplex Data....2-18
- Cyclic, Single Data....2-7
- Non-Cyclic, Multiplex Data....2-19
- Non-Cyclic, Single Data....2-9

Send to CLC
- Basic Example
  - Cyclic Data....2-7
  - Non-Cyclic Data....2-9
- Multiplex Example
  - Cyclic Data....2-17

Short Format 2....1-6
- Command Telegram Structure....4-7
- Messaging Example....4-13
- **Messaging Examples....4-8**
- Response Telegram Structure....4-7

Shutdown CLC (Fieldbus Error Reaction)....3-10

Single Data Types....1-3, 1-4

Source Object list
- Basic Example
  - Non-Cyclic Data....2-9

status word....1-5
- Multiplex Example
  - Cyclic Data....2-14

T

To FieldBus
- Basic Example....2-3
- Multiplex Example
  - Cyclic Data....2-14

Toggle Bit (T)....4-5

Transmission Format (Fmt)....4-5

V

View a Summary Report....3-1

VisualMotion ASCII Format1-6, **4-11**
- Command Telegram Structure....4-11
- Response Telegram Structure....4-12

W

Warning Only (Fieldbus Error Reaction)....3-10

X

X68 Connector.....5-3
X69 Connector.....5-3

Customer Service Locations

Germany

Sales area Center	Sales area East	Sales area West	Sales area North
INDRAMAT GmbH D-97816 Lohr am Main Bgm.-Dr.-Nebel-Str. 2 Telefon: 09352/40-4817 Telefax: 09352/40-4989	INDRAMAT GmbH D-09120 Chemnitz Beckerstraße 31 Telefon: 0371/3555-0 Telefax: 0371/3555-230	INDRAMAT GmbH D-40880 Ratingen Harkortstraße 25 Telefon: 02102/4318-0 Telefax: 02102/41315	INDRAMAT GmbH D-22525 Hamburg Kieler Str.212 Telefon: 040/853157-0 Telefax: 040/853157-15
Sales area South	**Sales area South-West**		**INDRAMAT Service-Hotline**
INDRAMAT GmbH D-80339 München Ridlerstraße 75 Telefon: 089/540138-30 Telefax: 089/540138-10	INDRAMAT GmbH D-71229 Leonberg Böblinger Straße 25 Telefon: 07152/972-6 Telefax: 07152/972-727		INDRAMAT GmbH Telefon: D-0172/660 040 6 -or- Telefon: D-0171/333 882 6

Customer service locations in Germany

Europe

Austria	Austria	Belgium	Denmark
G.L.Rexroth Ges.m.b.H. Geschäftsbereich INDRAMAT Hägelingasse 3 A-1140 Wien Telefon: +43 1/985 25 40-400 Telefax:+43 1/985 25 40-93	G.L.Rexroth Ges.m.b.H. Geschäftsbereich INDRAMAT Randlstraße 14 A-4061 Pasching Telefon: +43 7229/644 01-36 Telefax: +43 7229/644 01-80	Mannesmann Rexroth N.V.-S.A. Geschäftsbereich INDRAMAT Industrielaan 8 B-1740 Ternat Telefon: +32 2/582 31 80 Telefax: +32 2/582 43 10	BEC AS Zinkvej 6 DK-8900 Randers Telefon: +45 87/11 90 60 Telefax: +45 87/11 90 61
England	**Finnland**	**France**	**France**
Mannesmann Rexroth Ltd. INDRAMAT Division Broadway Lane, South Cerney Cirencester, Glos GL7 5UH Telefon: +44 1285/86 30 00 Telefax: +44 1285/86 30 03	Rexroth Mecman OY Riihimiehentie 3 SF-01720 Vantaa Telefon: +358 9/84 91 11 Telefax: +358 9/84 63 87	Rexroth - Sigma S.A. Division INDRAMAT Parc des Barbanniers 4, Place du Village F-92632 Gennevilliers Cedex Telefon: +33 1/41 47 54 30 Telefax: +33 1/47 94 69 41	Rexroth - Sigma S.A. Division INDRAMAT 17, Loree du Golf F-69380 Dommartin Telefon: +33 4/78 43 56 58 Telefax: +33 4/78 43 59 05
France	**Italy**	**Italy**	**Netherlands**
Rexroth - Sigma S.A. Division INDRAMAT 270, Avenue de lardenne F-31100 Toulouse Telefon: +33 5/61 49 95 19 Telefax: +33 5/61 31 00 41	Rexroth S.p.A. Divisione INDRAMAT Via G. Di Vittoria, 1 I-20063 Cernusco S/N.MI Telefon: +39 2/923 65-270 Telex: 331695 Telefax: +39 2/92 36 55 12	Rexroth S.p.A. Divisione INDRAMAT Via Borgomanero, 11 I-10145 Torino Telefon: +39 11/771 22 30 Telefax: +39 11/771 01 90	Hydraudyne Hydrauliek B.V. Kruisbroeksestraat 1a P.O. Box 32 NL-5280 AA Boxtel Telefon: +31 41 16/519 51 Telefax: +31 41 16/514 83
Spain	**Spain**	**Sweden**	**Switzerland**
Rexroth S.A. Centro Industrial Santiago Obradors s/n E-08130 Santa Perpetua de Mogoda (Barcelona) Telefon: +34 3/7 47 94 00 Telefax: +34 3/7 47 94 01	Goimendi S.A. División Indramat Jolastokieta (Herrera) Apartado 11 37 San Sebastion, 20017 Telefon: +34 43/40 01 63 Telex: 361 72 Telefax: +34 43/39 93 95	AB Rexroth Mecman INDRAMAT Division Varuvägen 7 S-125 81 Stockholm Telefon: +46 8/727 92 00 Telefax: +46 8/64 73 277	Rexroth SA Département INDRAMAT Chemin de l`Ecole 6 CH-1036 Sullens Telefon:+41 21/731 43 77 Telefax: +41 21/731 46 78
Switzerland	**Russia**		
Rexroth AG Geschäftsbereich INDRAMAT Gewerbestraße 3 CH-8500 Frauenfeld Telefon: +41 52/720 21 00 Telefax: +41 52/720 21 11	Tschudnenko E.B. Arsenia 22 153000 Ivanovo Rußland Telefon: +7 93/22 39 633		

European Customer service locations without Germany

Outside Europe

Argentina Mannesmann Rexroth S.A.I.C. Division INDRAMAT Acassusso 48 41/7 1605 Munro (Buenos Aires) Argentina Telefon: +54 1/756 01 40 +54 1/756 02 40 Telex: 262 66 rexro ar Telefax: +54 1/756 01 36	Argentina Nakase Asesoramiento Tecnico Diaz Velez 2929 1636 Olivos (Provincia de Buenos Aires) Argentina Argentina Telefon +54 1/790 52 30	Australia Australian Industrial Machinery Services Pty. Ltd. Unit ¾5 Horne ST Campbellfield VIC 2061 Australia Telefon: +61 3/93 59 0228 Telefax: +61 3/93 59 02886	Brazil Mannesmann Rexroth Automação Ltda. Divisão INDRAMAT Rua Georg Rexroth, 609 Vila Padre Anchieta BR-09.951-250 Diadema-SP Caixa Postal 377 BR-09.901-970 Diadema-SP Telefon: +55 11/745 90 65 +55 11/745 90 70 Telefax: +55 11/745 90 50
Canada Basic Technologies Corporation Burlington Division 3426 Mainway Drive Burlington, Ontario Canada L7M 1A8 Telefon: +1 905/335-55 11 Telefax: +1 905/335-41 84	China Rexroth (China) Ltd. Shanghai Office Room 206 Shanghai Intern. Trade Centre 2200 Yanan Xi Lu Shanghai 200335 P.R. China Telefon: +86 21/627 55 333 Telefax: +86 21/627 55 666	China Rexroth (China) Ltd. Shanghai Parts & Service Centre 199 Wu Cao Road, Hua Cao Minhang District Shanghai 201 103 P.R. China Telefon: +86 21/622 00 058 Telefax: +86 21/622 00 068	China Rexroth (China) Ltd. 1430 China World Trade Centre 1, Jianguomenwai Avenue Beijing 100004 P.R. China Telefon: +86 10/50 50 380 Telefax: +86 10/50 50 379
China Rexroth (China) Ltd. A-5F., 123 Lian Shan Street Sha He Kou District Dalian 116 023 P.R. China Telefon: +86 411/46 78 930 Telefax: +86 411/46 78 932	Hongkong Rexroth (China) Ltd. 19 Cheung Shun Street 1st Floor, Cheung Sha Wan, Kowloon, Honkong Telefon: +852 2741 13 51/-54 und +852 741 14 30 Telex: 3346 17 GL REX HX Telefax: +852 786 40 19 +852 786 07 33	India Mannesmann Rexroth (India) Ltd. INDRAMAT Division Plot. 96, Phase III Peenya Industrial Area Bangalore - 560058 Telefon: +91 80/839 21 01 +91 80/839 73 74 Telex: 845 5028 RexB Telefax: +91 80/839 43 45	Japan Rexroth Co., Ltd. INDRAMAT Division I.R. Building Nakamachidai 4-26-44 Tsuzuki-ku, Yokohama 226 Japan Telefon: +81 45/942-72 10 Telefax: +81 45/942-03 41
Korea Rexroth-Seki Co Ltd. 1500-12 Da-Dae-Dong Saha-Gu, Pusan, 604-050 Telefon: +82 51/264 90 01 Telefax: +82 51/264 90 10	Korea Seo Chang Corporation Ltd. Room 903, Jeail Building 44-35 Yoido-Dong Youngdeungpo-Ku Seoul, Korea Telefon: +82 2/780-82 07 ~9 Telefax: +82 2/784-54 08	Mexico Motorización y Diseño de Controles, S.A. de C.V. Av. Dr. Gustavo Baz No. 288 Col. Parque Industrial la Ioma Apartado Postal No. 318 54060 Tlalnepantla Estado de Mexico Telefon: +52 /397 86 44 Telefax: +52 /398 98 88	
USA Rexroth Corporation INDRAMAT Division 5150 Prairie Stone Parkway Hoffman Estates, Illinois 60192 Telefon: +1 847/645-36 00 Telefax: +1 847/645-62 01	USA Rexroth Corporation INDRAMAT Division 2110 Austin Avenue Rochester Hills, Michigan 48309 Telefon: +1 810/853-82 90 Telefax: +1 810/853-82 90	USA Rexroth Corporation INDRAMAT Division Northeastern Sales Office 7 Columbia Blvd. Peabody, MA 019660 Telefon: +1 508/531-25 74 Telefax: +1 508/531-2574	USA Rexroth Corporation INDRAMAT Division Southeastern Sales Office 3625 Swiftwater Park Drive Suwanee, GA 30174 Telefon: +1 770/932 3200 Telefax: +1 770/932-1903

Customer service locations outside Europe

Notes

282762

Printed in Germany

Rexroth Indramat